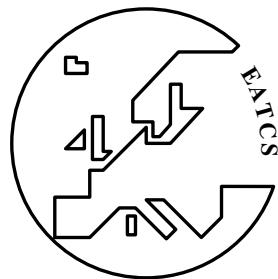


ISSN 0252-9742

Bulletin
of the
**European Association for
Theoretical Computer Science**

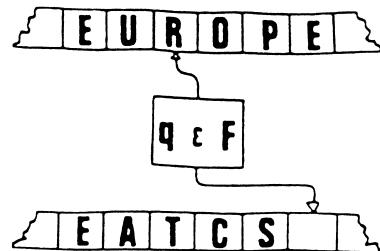
EATCS



Number 146

June 2025

**COUNCIL OF THE
EUROPEAN ASSOCIATION FOR
THEORETICAL COMPUTER SCIENCE**



PRESIDENT:

GIUSEPPE F. ITALIANO

ITALY

VICE PRESIDENTS:

ANTOINE AMARILLI

FRANCE

INGE LI GØRTZ

DENMARK

TREASURER:

GABRIELE FICI

ITALY

BULLETIN EDITOR:

STEFAN SCHMID

GERMANY

ANTOINE AMARILLI

FRANCE

THORE HUSFELDT

SWEDEN, DENMARK

IVONA BEZAKOVA

USA

GIUSEPPE F. ITALIANO

ITALY

TIZIANA CALAMONERI

ITALY

EMANUELA MERELLI

ITALY

THOMAS COLCOMBET

FRANCE

ANCA MUSCHOLL

FRANCE

ANNE DRIEMEL

GERMANY

CHARLES PAPERMAN

FRANCE

FUNDA ERGÜN

USA

EVA ROTENBERG

DENMARK

JAVIER ESPARZA

GERMANY

JIRI SGALL

CZECH REPUBLIC

GABRIELE FICI

ITALY

JUKKA SUOMELA

FINLAND

INGE LI GOERTZ

DENMARK

SZYMON TORUNCZYK

POLAND

FABRIZIO GRANDINI

SWITZERLAND

BIANCA TRUTHE

GERMANY

PAST PRESIDENTS:

MAURICE NIVAT (1972–1977)

MIKE PATERSON (1977–1979)

ARTO SALOMAA (1979–1985)

GRZEGORZ ROZENBERG (1985–1994)

WILFRED BRAUER (1994–1997)

JOSEP DÍAZ (1997–2002)

MOGENS NIELSEN (2002–2006)

GIORGIO AUSIELLO (2006–2009)

BURKHARD MONIEN (2009–2012)

LUCA ACETO (2012–2016)

PAUL SPIRAKIS (2016–2020)

ARTUR CZUMAJ (2020–2024)

SECRETARY OFFICE:

DMITRY CHISTIKOV

UK

EFI CHITA

GREECE

EATCS COUNCIL MEMBERS

EMAIL ADDRESSES

ANTOINE AMARILLI	A3NM@A3NM.NET
IVONA BEZAKOVA	IB@CS.RIT.EDU
TIZIANA CALAMONERI	CALAMO@DI.UNIROMA1.IT
THOMAS COLCOMBET	THOMAS.COLCOMBET@IRIF.FR
ANNE DRIEMEL	DRIEMEL@CS.UNI-BONN.DE
FUNDA ERGÜN	CHAIR.SIGACT@SIGACT.ACM.ORG
JAVIER ESPARZA	ESPARZA@IN.TUM.DE
GABRIELE FICI	GABRIELE.FICI@UNIPA.IT
INGE LI GOERTZ	INGE@DTU.DK
FABRIZIO GRANDONI	FABRIZIO@IDSIA.CH
THORE HUSFELDT	THORE@ITU.DK
GIUSEPPE F. ITALIANO	GIUSEPPE.ITALIANO@UNIROMA2.IT
EMANUELA MERELLI	EMANUELA.MERELLI@UNICAM.IT
ANCA MUSCHOLL	ANCA@LABRI.FR
CHARLES PAPERMAN	CHARLES.PAPERMAN@UNIV-LILLE.FR
EVA ROTENBERG	EVA@ROTHENBERG.DK
STEFAN SCHMID	STEFAN.SCHMID@TU-BERLIN.DE
JIRI SGALL	SGALL@IUUK.MFF.CUNI.CZ
JUKKA SUOMELA	JUKKA.SUOMELA@AALTO.FI
SZYMONTORUNCGZYK	SZYMATOR@mimuw.edu.pl
BIANCA TRUTHE	BIANCA.TRUTHE@INFORMATIK.UNI-GIESSEN.DE

Bulletin Editor: Stefan Schmid, Berlin, Germany
Cartoons: DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by PDF_TE_X and Con_TE_XT in TXFONTS.

All contributions are to be sent electronically to

bulletin@eatcs.org

and must be prepared in L_AT_EX₂_E using the class beatcs.cls (a version of the standard L_AT_EX₂_E article class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Please, consult <http://www.eatcs.org/bulletin/howToSubmit.html>.

We regret we are unfortunately not able to accept submissions in other formats, or indeed submission not *strictly* adhering to the page and font layout set out in beatcs.cls. We shall also not be able to include contributions not typeset at camera-ready quality.

The details can be found at <http://www.eatcs.org/bulletin>, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email bulletin@eatcs.org.

Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

The EATCS home page is <http://www.eatcs.org>

Table of Contents

EATCS MATTERS

LETTER FROM THE PRESIDENT	3
LETTER FROM THE EDITOR	7

EATCS COLUMNS

TCS ON THE WEB, *by S. Neumann*

BRINGING THEORETICAL COMPUTER SCIENCE TO YOUTUBE, A CONVERSATION WITH TIM ROUGHGARDEN, <i>by S. Neumann</i>	13
---	----

THE DISTRIBUTED COMPUTING COLUMN, *by S. Gilbert*

RECENT RESULTS IN DAG BFT, <i>by N. Giridharan</i>	19
--	----

THE FORMAL LANGUAGE THEORY COLUMN, *by G. Pighizzini*

GENERALIZED DIRECTIONS ON A COMPASS, <i>by T.A. Hartmann</i>	43
--	----

THE EDUCATION COLUMN *by D. Komm and T. Zeume*

ASSESSING GPT PERFORMANCE IN A PROOF-BASED UNIVERSITY-LEVEL COURSE UNDER BLIND GRADING, <i>by</i> <i>M. Ding, F. Soldà, W. Yuan, R. Kyng</i>	57
--	----

THE LOGIC IN COMPUTER SCIENCE COLUMN, *by Y. Gurevich*

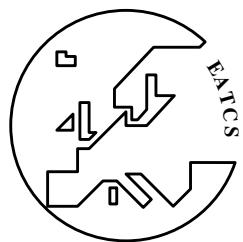
INTERACTIVE CLASSICAL ALGORITHMS: PREVIEW, <i>by</i> <i>Y. Gurevich</i>	83
--	----

NEWS AND CONFERENCE REPORTS

REPORT FROM EATCS JAPAN CHAPTER, <i>by Y. Yamauchi</i>	101
--	-----

EATCS LEAFLET	106
---------------------	-----

EATCS Matters





Dear EATCS members,

As the vibrant energy of summer 2025 unfolds, I extend my warmest greetings and heartfelt wishes to each and every one of you. I hope this message finds you well and thriving, immersed in the fascinating world of theoretical computer science.

The anticipation is building, as the June issue of our Bulletin is poised for release, arriving just in time for our flagship event: ICALP 2025. Like every year, our annual general assembly will be held alongside the conference, and I sincerely hope that many of you will join us in the beautiful city of Aarhus, Denmark, from July 8th to 11th.

Ioannis Caragiannis, Kasper Green Larsen, and their dedicated team at the University of Aarhus have poured their hearts and minds into crafting what promises to be a truly exceptional ICALP. Their tireless efforts are ensuring a scientifically stimulating and personally enriching experience for all attendees. I am confident that this will be a truly memorable event, brimming with opportunities for discovery and connection.

Beyond the compelling invited and contributed talks, ICALP 2025 will be a moment to celebrate the extraordinary achievements within our community. We are immensely proud to announce the recipients of this year's prestigious awards:

- *The EATCS Award 2025 will be presented to the distinguished Rajeev Alur.*
- *The Presburger Award 2025 will honor*



the remarkable contributions of Sepehr Assadi and Tomasz Kociumaka.

- *The EATCS Distinguished Dissertation Award will recognize the outstanding doctoral research of Joakim Blikstad, Christoph Grunau, and Tuukka Korhonen. Furthermore, we will have the distinct pleasure of inducting the EATCS Fellows Class of 2025, celebrating individuals who have made profound and lasting impacts on our field:*

- *Kasper Green Larsen, for his "outstanding contributions in many areas of theoretical computer science, including the theory of machine learning, data structures, lower bounds, and cryptography."*
- *Anca Muscholl, for her "seminal contributions to the study of automata and logic and for her outstanding service to the community."*

My deepest gratitude goes to the members of the award and fellow committees for their diligent and thoughtful work in selecting this stellar group of awardees and fellows. It will be a tremendous honor to collectively celebrate their inspiring work and dedication during ICALP 2025.

I also warmly invite you to attend the EATCS General Assembly, which will take place on Wednesday evening during the ICALP week in Aarhus. This is a vital opportunity to gain detailed insights into the EATCS's activities over the past year and our exciting plans for the year ahead. Your presence and



engagement are invaluable as we shape the future of our association.

I eagerly anticipate seeing many of you in Aarhus, not only to share in the intellectual vibrancy of ICALP 2025 but also to engage in meaningful discussions at the general assembly about how we can collectively enhance the EATCS impact within the computer science community.

As always, please know that my door, or rather, my inbox, is always open. Your comments, constructive criticisms, and innovative suggestions are not just welcome, but actively sought. We are a community, and your insights are crucial in helping us refine our efforts and maximize the EATCS's positive influence on our beloved field. Every suggestion will be carefully considered.

I genuinely look forward to connecting with you, whether in Aarhus or through your thoughtful correspondence.

With warm regards and anticipation,

*Giuseppe F. Italiano
Luiss University, Rome
President of EATCS
president@eatcs.org*

June 2025

BEATCS no 146



Dear EATCS member!

I hope you're enjoying the first days of summer! Even here in the far north—at a workshop at NTNU in Trondheim—faculty and students are celebrating the end of the semester and exams, with warm evenings that stretch late into the night.

It's the perfect time to head outside, unwind, and perhaps enjoy the latest issue of the Bulletin!

In this edition:

In the TCS on the Web column, Stefan Neumann interviews Tim Roughgarden about the origins of his YouTube channel, the teaching philosophy behind his videos, and how to make theoretical ideas accessible to a broader audience.

In the Distributed Computing column, Neil Giridharan surveys exciting recent work on DAG Byzantine fault-tolerant protocols, including Bullshark—the first partially synchronous DAG BFT protocol—and how newer protocols like Shoal++ build upon it.

In the Formal Language Theory column, Tim A. Hartmann explores recursive naming procedures underlying generalized compass directions, linking them to tabled Lindenmayer systems, which are typically used to model plant growth.

In the Education column, Ding and colleagues at ETH Zurich evaluate the performance of GPT-4o and o1-preview in a proof-based university course, under blind grading.

In the Logic in Computer Science column, Yuri Gurevich presents a thought-provoking

BEATCS no 146

*dialogue on the axiomatization of
interactive classical algorithms.*

*You'll also find a report from the EATCS
Japan Chapter in this issue.*

Enjoy reading-and have a wonderful summer!

Warm regards,

*Stefan Schmid, Berlin
June 2025*



Institutional Sponsors

BEATCS no 146

CTI, Computer Technology Institute & Press "Diophantus"
Patras, Greece

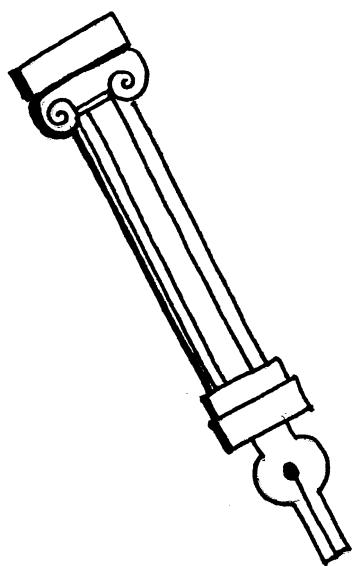
CWI, Centum Wiskunde & Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Columns



TCS ON THE WEB

BY

STEFAN NEUMANN

TU Wien
Erzherzog-Johann-Platz 1
1040 Vienna, Austria
stefan.neumann@tuwien.ac.at
<https://neumannstefan.com>

Tim Roughgarden is a full professor at Columbia University. He works on algorithmic game theory and microeconomics, especially as applied to networks, auctions, and blockchains. He has made significant contributions to the design and analysis of algorithms, and his work has won many awards, including the 2009 Grace Murray Hopper Award, the 2012 Gödel Prize, and the 2016 Kalai Prize.

Tim is one of the most important theoretical computer scientists on YouTube¹, where he regularly uploads recordings of his lectures. He has more than 25,000 followers, and his videos have been watched more than 1 million times in total. In this column, he answers our questions about the origins of his YouTube channel, the teaching philosophy behind his videos, and how to communicate theoretical ideas to a broader audience.

¹<https://www.youtube.com/@timroughgardenlectures1861>

BRINGING THEORETICAL COMPUTER SCIENCE TO YOUTUBE

A Conversation with Tim Roughgarden

Tim, thank you for taking the time for this interview. You created your YouTube channel back in 2013. Do you recall what initially motivated you to start uploading content?

First, let me say that it's an honor to do this interview. I've been reading the EATCS Bulletin — especially the excellent survey papers that have been published in it — since my days as a PhD student.

As to your question, the story begins in 2011, when there was tremendous appetite at Stanford (where I was at the time) for experimentation in online education. The companies Coursera and Udacity were both formed around this time by Stanford colleagues, and I pounced on the opportunity to develop my long-running undergraduate algorithms course into one of the first "MOOCs" (massive online open courses) on Coursera. The response was staggering, with hundreds of thousands of learners signing up for it. In addition to the sheer numbers, I was struck by how many of the learners were hungry for high-quality technical content but in no position to attend the exclusive schools where that content is traditionally taught (due to geography, family obligations, financial resources, etc.). To this day, I regularly receive emails from alumni of my online courses thanking me for the personal and professional advances that those courses unlocked for them. Since these experiences, I've been on a mission to make as many educational resources as widely available as possible.

You used to upload videos of board lectures of 60–90 minutes length. More recently, your videos are shorter—of about 30 minutes length—and now you write your content on a tablet (but still by hand). What has triggered that change and why do you not use slides?

To my taste, the gold standard for a video is a simulation of a one-on-one session with an expert tutor. The shorter videos in which I'm writing by hand are my best attempt at (a noninteractive version of) such a simulation. They are time-consuming to produce and have never contributed to my official teaching duties at Stanford or Columbia, so I can put them out only as time allows. The longer

videos at the white/blackboard are all of in-person courses that I delivered at Stanford to fulfill my teaching duties there. To all the instructors out there, especially if you’re teaching a topic that doesn’t already have adequate coverage on YouTube, I strongly encourage recording your lectures and making them public (with the permission of your employer). It increases your audience by something like 10–100x while increasing your workload by a far smaller amount (especially if you can enlist a teaching assistant or PhD student to assist with the video recording and production).

For theoretical computer science courses (and mathematical courses in general), as both a teacher and a student, I’ve always found white/blackboard lectures far more effective and engaging than slide-based ones. There’s something about the way you can talk through math as you’re writing it that seems quite difficult to simulate with a slide deck, no matter how much time you spend on your animations.

What do you think makes your lectures so popular?

That’s really a question for others! I’m just happy that they seem to have been helpful to so many different people.

Beyond your research and YouTube channel, your lecture notes¹ are also highly influential in our community. After you and Greg Valiant taught “Modern Algorithmic Toolbox” at Stanford, I noticed many instructors — including myself — started teaching consistent hashing. How did you succeed in popularizing your notes, and what do you think distinguishes them from others?

The demand for high-quality and free lecture notes is high, and the supply is low (presumably because there are no short-term incentives for creating them). So if you’ve got a lecture or a course that you’re proud of and that complements the free resources that are available already, by all means write it up and get it out there!

Sometimes my impression is that in TCS we have a lot of great results that would deserve more visibility in the general CS community. Do you have some suggestions to other TCS researchers on how we can improve our outreach?

I actually think that the TCS community has been quite strong on outreach for the past decade-plus (for example, through the programming at the Simons Institute) and is generally very well respected across computer science. For TCS readers looking to increase the audience of their own research, my general advice would be to: (i) identify the audience that you’d like to speak to (mathematicians? computer systems researchers? economists?); (ii) understand the typical

¹<https://timroughgarden.org/notes.html>

background of that audience and what it tends to care about (mathematical depth? practical impact? policy implications?); and (iii) communicate your work to that audience accordingly. For example, if you'd like your work to attain visibility in two separate research communities (e.g., "traditional" TCS as well as some other area), it is sometimes worth writing two versions of the same paper (e.g., a formal TCS publication as well as an informal writeup or blog post that is optimized for the second audience). Also, give as many talks and seminars as you can!

Before we conclude the interview, could you please let us know what your current research is about?

Most of my research over the past 25 years resides at the interface of computer science and economics (a.k.a. "algorithmic game theory (AGT)" or "economics and computation (econ-CS)"). Many problems central to modern computer science, ranging from resource allocation in large networks to the design of blockchain protocols, fundamentally involve interactions between multiple self-interested parties. Economics and game theory offer a host of useful models and definitions to reason about such problems. The flow of ideas also travels in the other direction, with approximation and complexity notions playing an increasingly important role in the latest developments in economic theory.

Working in algorithmic game theory has led me on a journey across a number of application domains, such as network routing and design, online advertising, spectrum auctions, and data-driven mechanisms. For the last several years, I've been working full-time on developing foundations for blockchain protocols and the applications built on top of them — I've never seen a technology in which important incentive issues are so deeply baked into the technology itself! The three main thrusts of my research in the area thus far (done with a number of fantastic collaborators) have been: (i) a general theory of permissionless consensus protocols; (ii) a new branch of mechanism design that is tailored to the unique constraints of blockchain protocols; and (iii) mathematical foundations for decentralized exchanges.

Finally, is there anything else you want our readers to know?

I'm always happy to receive feedback on any of my videos/books/lecture notes/papers/etc., as well as suggestions about which topics would be most useful to cover in the future!

Thank you for this nice interview, Tim!

The Bulletin of the EATCS

THE DISTRIBUTED COMPUTING COLUMN

Seth Gilbert

National University of Singapore

seth.gilbert@comp.nus.edu.sg

This month, in the Distributed Computing Column, Neil Giridharan surveys some exciting recent work on DAG BFT protocols. Over the last decade, Byzantine fault-tolerant (BFT) state machine replication has proved to be one of the most effective and robust general techniques for building a distributed service. Traditionally, such protocols were based around repeatedly executing Byzantine agreement, where servers agreed iteratively on the sequence of actions to take. More recently, however, there has been a new “DAG-based” approach where servers collectively construct a DAG that can then be ordered in a consistent manner. This new approach has led to significant performance improvements, and enabled a variety of new optimizations (e.g., decoupling data transfer from agreement).

In this survey, Neil Giridharan provides an overview of this new class of protocols. He begins by describing Bullshark, the first partially synchronous DAG BFT protocol; and he then shows how more recent protocols such as Shoal++, Sailfish, and Cordial Miners, Mysticeti, and Autobahn build on Bullshark to optimize performance.

Overall, this article provides a comprehensive overview of the state-of-the-art for DAG BFT protocols today, along with some nice insights into the subtleties involved in high performance state machine replication.

The Distributed Computing Column is particularly interested in contributions that propose interesting new directions and summarize important open problems in areas of interest. If you would like to write such a column, please contact me.

RECENT RESULTS IN DAG BFT

Neil Giridharan

UC Berkeley

Abstract

Traditional Byzantine fault tolerant (BFT) state machine replication (SMR) protocols have a reputation of having poor performance despite decades of work in optimizing these protocols. Recently, a new family of protocols, DAG BFT, has been shown to have much higher throughput than traditional BFT protocols despite having higher communication complexity. This has attracted a lot of interest in improving DAG BFT protocols. This brief survey gives background on the foundational DAG BFT protocols and summarizes the techniques of state-of-the-art results. Finally, it concludes by highlighting avenues for future work.

1 Introduction

Byzantine fault tolerant (BFT) state machine replication (SMR) protocols implement the abstraction of a single totally ordered log of client requests. These protocols must handle malicious failures and network asynchrony, which can arbitrarily delay messages. BFT SMR is a core building block of many systems such as blockchains, distributed databases, and cloud infrastructure.

The most well-known and extensively studied BFT SMR protocols are in the *traditional BFT* [6, 14, 26, 18, 11, 13, 15] family. Traditional BFT protocols typically work by having a leader drive progress, while having a view change mechanism to replace leaders that are faulty or slow. These family of protocols have been extensively optimized to achieve optimality in theoretical metrics such as round and communication complexity. Despite this theoretical optimality, traditional BFT SMR protocols have shown to have poor performance in practice [12, 7, 23].

Recently, a new family of BFT SMR protocols emerged known as *DAG BFT* [16, 7, 5, 10, 23, 22, 3, 17, 4]. These protocols work by having all replicas build a DAG of data proposals, and then totally ordering this DAG according to some deterministic rule to reach a consistent state. DAG BFT is suboptimal in theoretical metrics like communication complexity. However, they have impressive practical performance, achieving 10x throughput compared to traditional BFT protocols [7]. This

is in large part because they decouple data dissemination from agreement, alleviating a key throughput bottleneck of traditional BFT SMR.

These impressive performance results have attracted a lot of interest in DAG BFT from academics and industry practitioners alike. They have been adopted by several blockchain projects [5, 1, 10] and have been cited numerous times. Unfortunately, these protocols are not the easiest to understand. DAG BFT uses different techniques and arguments, making it difficult for those unfamiliar to develop a strong intuition for these protocols. Furthermore, it is also hard to compare DAG BFT protocols, and to understand the various trade-offs that each protocol makes. This makes it difficult to tweak and extend these protocols, which is often necessary when trying to deploy these protocols into production.

This paper is a survey on DAG BFT protocols in the partial synchrony model [9], as partial synchrony is the most widely used network model for BFT SMR. Our goals are twofold. We first hope to give an intuition of how DAG BFT works, so that unfamiliar readers can understand new DAG BFT protocols. Secondly, we aim to cover the various techniques of recent state-of-the-art DAG BFT protocols with the goal of illustrating the different trade-offs of each protocol.

We first give background on Bullshark [23, 24], the first partially synchronous DAG BFT protocol (Section §3). Understanding Bullshark is the key to understanding many of the newer DAG BFT protocols, which either directly use Bullshark in a black-box manner or modify its internals. Next, we begin to cover recent DAG BFT results starting with protocols in the *certified DAG* family (Section §4). These protocols utilize techniques that do not modify or change the Bullshark’s DAG structure. After that, we cover protocols in the *uncertified DAG* family (Section §5). These protocols use a similarly structured DAG but with different properties. Finally, we cover protocols in the *hybrid DAG* family (Section §6, which combine characteristics of traditional leader-based BFT protocols with DAG BFT).

2 Preliminaries

All covered protocols adopt the most popular BFT system model [6]. We assume a total of $n = 3f + 1$ replicas, where at most f of the n replicas can be faulty. Faulty replicas can behave arbitrarily, while correct replicas must execute the protocol faithfully. We assume a static adversary that can coordinate the actions of faulty replicas. Partial synchrony [9] is the adopted network model. In partial synchrony, there exists an unknown Global Stabilization Time (GST) and a known upper bound Δ on the message delivery time, such that before GST the network is asynchronous (no bound on message delivery time), but after GST all messages are guaranteed to arrive within Δ . We also often refer to δ as the *ac-*

tual message delay when analyzing latency. We assume standard cryptographic primitives like hash functions and digital signatures and assume that they cannot be broken by the adversary. Replicas use authenticated, reliable, point-to-point channels to communicate. They are assumed to check the validity of signatures before processing the corresponding messages.

BFT SMR outputs a linearizable totally ordered log of client requests which satisfy the following properties:

- **Safety.** No two correct replicas commit different values at the same log position.
- **Liveness.** All client requests are eventually committed in the log.
- **External Validity.** Any committed value must satisfy an external predicate P .

3 Bullshark

In this section, we give background on the Bullshark [23, 24] protocol, since it was the first partially synchronous DAG BFT protocol. Bullshark consists of two core components: (i) an algorithm that constructs a DAG of data proposals, and (ii) an algorithm that outputs a consistent total order of the DAG. We first focus on the algorithm that constructs the DAG (§3.1). We then show how the ordering algorithm (§3.2) totally orders the DAG without exchanging any additional messages.

3.1 Constructing the DAG

The DAG is divided into a number of integer-valued *rounds* (starting from 0), where in each round there is at most one vertex per replica. A *vertex* in the DAG represents a data proposal from a source replica, while an *edge* indicates a causal relationship between vertices. A valid vertex in round r only contains edges to at least $n - f$ vertices from round $r - 1$. The *causal history* of a vertex, v , is the set of vertices that are reachable from v by following some sequence of edges. In every even-numbered round there is a pre-defined leader. An *anchor* is a vertex whose source replica is the leader for the round. The ordering algorithm uses anchors to reach a consistent total order.

DAG properties. The ordering algorithm requires the DAG to satisfy the following properties:

- **Non-equivocation.** For any position in the DAG, no two correct replicas will have different vertices.
- **Data availability.** For any vertex in a correct replica’s DAG, at least one correct replica has stored the corresponding data proposal.

A DAG that satisfies these properties is known as a *certified* DAG.

Adding Vertices to the DAG. To add a vertex to the DAG, a replica invokes *consistent broadcast* on its data proposal. Consistent broadcast guarantees the following properties:

- **Validity.** If the source replica is honest, then all correct replicas will deliver the source replica’s vertex.
- **Consistency.** If a correct replica delivers vertex v and another correct replica delivers vertex v' , then $v = v'$.
- **Integrity.** A correct replica delivers at most vertex.

Consistency ensures the non-equivocation property of the DAG, while validity and integrity ensure that each DAG round contains at least $n - f$ vertices — one per replica. When a replica delivers a vertex from consistent broadcast it obtains a certificate, which acts as a proof of delivery and a proof of data availability. Then it adds the certificate to its local DAG, so that its next vertex can include at least $n - f$ certificates.

Consistent Broadcast Protocol. When a replica, r_i has a new vertex, v_i , containing a data proposal and $n - f$ certificates (edges) it sends a signed PROPOSE message containing v_i to all replicas. Upon receiving a PROPOSE message from a replica, r_i , a replica r_j checks (i) that the vertex contains $n - f$ certificates from the previous round and (ii) that it has not voted for another vertex in the same round from the same source replica. If these checks pass, a replica stores the associated data proposal, and sends a signed VOTE message containing a hash of the vertex to replica r_i . When replica r_i receives $n - f$ matching VOTE messages, it assembles them into a certificate, acting as a proof of delivery. Replica r_i then sends this certificate to all replicas. Once a replica receives this certificate it delivers vertex v_i and adds the certificate to its DAG.

Advancing DAG rounds. Once a replica has delivered $n - f$ vertices in round r , it has enough information to create a new vertex for round $r + 1$. However, eagerly doing so can compromise liveness of the ordering algorithm, which relies on hearing from slow but correct replicas. As a result, a replica starts a timer in every DAG round to wait for vertices from slow but correct replicas.

In even rounds, a replica waits to hear from the leader. It can move to the next round if it delivers the anchor or if the timer expires. For odd rounds, a replica can advance if the ordering algorithm commits the anchor in the previous round or if the timer expires. As an optimization (in odd rounds), replicas can also advance if they conclude that it was impossible for the anchor to be committed.

3.2 Totally Ordering the DAG

In traditional state machine replication or atomic broadcast protocols, replicas agree on an ordered sequence of values. Bullshark takes a similar approach by agreeing on a sequence of anchors. Once a sequence of anchors is committed, the causal history of each anchor in the sequence is ordered by any deterministic rule, to establish a consistent total order for all vertices in the DAG. Thus, the main goal of the ordering algorithm is to agree on a sequence of anchors. An example execution of the ordering algorithm is shown in Figure 1.

The sequence of anchors can be thought of a hash chain, where each anchor has a parent anchor. The parent of an anchor, A_i , is defined as the anchor with the highest round in A_i 's causal history. An anchor A_j extends another anchor A_i if A_j is a descendant of A_i in the hash chain. Given an anchor, A , the sequence of anchors is determined by recursively computing the parent for each anchor. Once a replica commits an anchor, all prior anchors in the hash chain are also committed. To preserve safety, the commit rule needs to satisfy the following invariant.

- **Extension.** If a correct replica commits an anchor, A , in round r , then any anchor in round $r' > r$ must extend A .

The extension invariant ensures safety as any committed anchor must be in the prefix of any later anchor that is committed. For example, if one correct replica commits anchor A_1 and another correct replica does not (due to asynchrony or faults), then once the other correct replica commits a future anchor, A_2 , A_2 is guaranteed to extend A_1 ; thus, A_1 will also be committed.

Commit Rule. The commit rule counts the number of votes for an anchor, where a vote is a vertex that has an edge to the anchor. A replica can commit an anchor, A , in round r if it observes $f + 1$ vertices (votes) in round $r + 1$ that

each have an edge to A . This guarantees that any vertex in round $r + 2$ will have an edge to at least one of the $f + 1$ votes (by quorum intersection). The extension property is satisfied, since any future anchor is in round $r + 2$ or contains some vertex in round $r + 2$, which must have a path to the anchor in round r . Note that the commit threshold only needs to be $f + 1$ rather than $2f + 1$, since the DAG guarantees non-equivocation.

Final Total Order. Recall that an anchor, A , defines a hash chain of anchors where A is the tip. Once an anchor is committed, all anchors in the hash chain are also committed. To compute the anchor sequence, a replica recursively gets the parent anchor, which is the latest anchor in its causal history.

Once the sequence of anchors is computed, a replica will iterate through each anchor in order and execute its causal history in some deterministic order. This is safe since the causal history of an anchor is unique.

3.3 Analysis

Communication Complexity. Each vertex has size $O(n)$, since it contains $n - f$ edges. Each edge is a certificate containing $O(n)$ signatures, but assuming threshold signatures this cost can be reduced to $O(1)$. Consistent broadcast has $O(n^2)$ complexity for messages of size $O(n)$. So since there are $O(n)$ replicas each broadcasting a vertex, the total complexity is $O(n^3)$.

Throughput. Communication complexity has often been assumed to negatively correlate with throughput. However, prior work [7] observed that in practice data dissemination is a major throughput bottleneck. Bullshark parallelizes data dissemination across all replicas, avoiding a bottleneck at a single leader replica unlike other traditional BFT protocols. This results in better bandwidth utilization and higher throughput compared to traditional BFT protocols [7, 23], despite the higher communication complexity. In practice with batching, consensus metadata messages are often much smaller in size compared to data proposals. Thus, the overhead from more metadata messages does not significantly impact throughput.

Latency. We analyze latency under good-case conditions when the network is synchronous, and there are no faults. Anchors take 2 DAG rounds to commit, where each DAG round takes 3δ (message delays) due to consistent broadcast. Vertices in odd (non-anchor) rounds require an additional DAG round to be committed, since they must wait to be included by the anchor in the next round. Finally, vertices in even (anchor) rounds that are not the anchor require 4 DAG

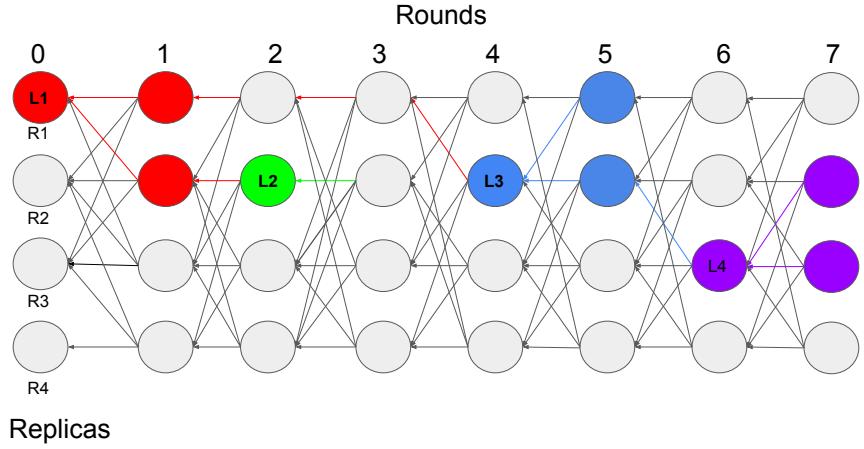


Figure 1: Illustration of the DAG at replica R_1 with $n = 4, f = 1$. The columns represent the round numbers and the rows are all the vertices from a particular source replica. L_1 denotes the first anchor (in round 0). There are $f + 1 = 2$ votes for L_1 , so L_1 is committed. L_2 is the second anchor in round 2. L_1 is the parent of L_2 in the hash chain sequence since it is the latest anchor in L_2 's causal history. L_2 only has one vote for it, so it is not committed. The next anchor, L_3 , does not have L_2 in its causal history, but does have a path to L_1 , so L_1 is its parent anchor. There are 2 votes for L_3 in round 5, so the hash chain anchor sequence L_1, L_3 is committed. Anchor L_4 has L_3 in its causal history, so L_3 is its parent anchor. Similarly, it has 2 votes and so the hash chain anchor sequence L_1, L_3, L_4 is committed.

rounds to commit, since they must wait until the next anchor (in the next even round) includes them.

In summary, for an even rounds the anchor has a commit latency of 6δ , while non-anchor vertices have a commit latency 12δ . For odd rounds, vertices have a commit latency of 9δ . Thus, the expected commit latency is 10.5δ . However, the full consensus latency encapsulates not just the commit latency but also the queuing latency for a client request to be included in a vertex. Since a vertex is disseminated every DAG round, and each DAG round takes 3δ , on average it will take 1.5δ for a request to be included in a vertex. Thus, the full consensus latency is 12δ .

Chain Quality. In traditional BFT protocols there is no bound on the number of committed proposals from Byzantine replicas. However, in Bullshark, each DAG round contains at least $n - f$ vertices, of which at least a majority are from correct replicas. This guarantees a property called *chain-quality* which states that at least $\frac{1}{2}$ of committed blocks must be from correct replicas.

Fairness. The version of Bullshark described here lacks fairness in the sense that vertices from some correct replicas may be "orphaned". Since each vertex is only guaranteed to have $n - f$ edges, vertices from slow but correct replicas in round r may not be referenced by vertices in round $r + 1$. Once this happens, the orphaned vertices from round r are effectively abandoned since no future vertex will reference them. To solve this issue, Bullshark adopts weak edges [16] to allow vertices to also have edges to some vertices not in the immediate prior round. Weak edges make garbage collection difficult, since vertices can reference prior vertices in arbitrarily lower rounds. We omit discussing how to do garbage collection while preserving fairness and refer to the Bullshark paper [23] for more details.

Data Synchronization. For a given vertex, v , a replica may only have a certificate proving data availability, and not the actual corresponding data payload for v or for vertices in v 's causal history. A replica therefore must synchronize with other replicas to fetch all payload data if it does not already have the data locally. This data synchronization process unfortunately is sequential and may require multiple network round-trips. To see why, take an example of a vertex v in round 3. First, a replica must fetch the data for v 's edges in round 2. Only when the vertices from round 2 are fetched does a replica know what vertices to synchronize on from round 1. Similarly, once all data for the vertices in round 1 are fetched, the replica must synchronize on the data for vertices in round 0.

Key question. Of the metrics we analyzed, latency is the major one for which Bullshark is suboptimal compared to other BFT protocols. Bullshark's consensus latency of 12δ is significantly higher than the optimal 3δ that leader-based protocols can achieve [2]. Thus, most recent work address the key question of whether it is possible to design a DAG BFT protocol that achieves low latency. Recent work has answered this question in the affirmative, using different designs. We classify the recent results into three different protocol families: Certified DAG, Uncertified DAG, and Hybrid DAG, and start by discussing the Certified DAG family.

4 Certified DAG

In this section, we cover the Shoal [22], Shoal++ [3], and Sailfish [21] protocols. These protocols belong to the certified DAG family. The certified DAG family of protocols adopt the same DAG as in Bullshark but adopt various techniques to reduce Bullshark’s latency.

4.1 Shoal

Shoal [22] does not modify Bullshark internals, but instead applies two general-purpose techniques to reduce latency: pipelining and leader reputation. Shoal takes as a starting point a single-shot version of the Bullshark protocol. Single-shot Bullshark runs on a certified DAG starting at some round r . It terminates immediately upon committing the first anchor in some round $r' \geq r$.

Pipelining. Shoal creates a pipelining effect by invoking multiple single-shot Bullshark instances sequentially. Once an anchor is committed in round r , a new instance is started in round $r + 1$. Optimistically, every instance will commit an anchor in the first round it was invoked, resulting in an anchor being committed in every round. As in Bullshark, once an anchor is committed its causal history is ordered in some deterministic way. Safety is preserved since each instance inherits Bullshark’s safety property, which ensures all correct replicas will agree on the same first anchor to order. Bullshark’s safety property also ensures all correct replicas will agree on which round to invoke each new single-shot Bullshark instance.

Leader reputation. The pipelining effect can be nullified by faulty or slow leaders. To see this, suppose the first single-shot Bullshark instance commits an anchor in round 0 but the second instance commits an anchor in round 3 instead of round 1 because the round 1 leader was faulty. Vertices in rounds 1 and 2 experience high latency because they have to wait for the anchor in round 3 to be committed. To mitigate this issue, replicas can assign reputation scores to other replicas, and only elect leaders with high reputation scores. These reputation scores can be computed based on past participation history. A key challenge is ensuring that all correct replicas agree on reputation scores; otherwise, different replicas can elect different leaders when running a single-shot Bullshark instance, causing a possible safety violation. Shoal solves this challenge by computing reputation scores deterministically using only the causal history of the last committed anchor. Since all correct replicas will agree on the same first anchor to commit, and use the

same deterministic function, they will compute the same reputation scores, and thus compute the same leaders to use for the next single-shot Bullshark instance.

4.2 Shoal++

Shoal still has the same commit latency for anchors as Bullshark (6δ). Shoal++ [3] proposes a faster commit rule to reduce this latency down to 4δ . Additionally, Shoal++ more aggressively pipelines to further reduce consensus latency.

Fast Commit Rule. Recall that the Bullshark commit rule requires $f + 1$ votes delivered through consistent broadcast. Shoal++ makes the observation that receiving $2f + 1$ PROPOSE messages for vertices in round $r + 1$ that have an edge to the anchor in round r is enough to commit the anchor in round r (without needing to wait for consistent broadcast to finish for those vertices). To see why, note that of these $2f + 1$ messages at least $f + 1$ are from correct replicas. By the validity property of consistent broadcast, all correct replicas are guaranteed to eventually deliver the vertex that appears in a PROPOSE message from a correct source replica. Thus, since there are at least $f + 1$ votes that will eventually be delivered for the anchor, it is safe to commit without waiting for consistent broadcast to complete. This saves two message delays compared to Bullshark’s commit rule, reducing the anchor commit latency to 4δ .

Aggressive Pipelining. Shoal limited pipelining to at most one new single-shot Bullshark instance per round. Shoal++ supports pipelining multiple single-shot instances in the same round as in Mysticeti [4] (covered in Section §5.2). Shoal++ pipelines up to n instances in the same round. Each instance is assigned a monotonically increasing sequence number (starting from 1) corresponding to a position in a totally ordered log as in PBFT [6].

Unlike in Shoal, each single-shot instance is initialized with a round r and a source replica i . Replica i ’s vertex in round r acts as the first anchor in the single-shot instance. The instance either decides to commit replica i ’s vertex in round r or to skip it. Once a correct replica has committed or skipped an instance with sequence number j , it waits until all sequence numbers from 1 through $j - 1$ have been committed or skipped. It then iterates through each sequence number in order and either executes the causal history of the committed vertex in some deterministic order or skips the instance. By the safety property of Bullshark, all correct replicas will agree on the same committed vertex or all agree to skip, so a consistent total order is established. This aggressive pipelining essentially makes all vertices anchors in some instance, and so with the fast commit rule they experience a commit latency of 4δ .

More DAGs. Shoal++ additionally pipelines more DAGs to reduce the queuing delay for a client request to be included in a vertex. Normally a client request would take on average 1.5δ to be included in a vertex, since consistent broadcast completes in 3δ . Shoal++ staggers three DAGs in parallel, that are each offset by one message delay, so that for a given replica, a vertex is proposed every message delay in some DAG. Therefore, a new client request can be included in the next vertex of whichever DAG is ready, reducing the queuing delay to 0.5δ . To maintain a consistent total order, the committed anchor sequence for each DAG is interleaved into a single totally ordered log, where the causal history of each anchor is executed in some deterministic order.

4.3 Sailfish

Unfortunately, the fast commit rule is not always guaranteed to trigger in good-case conditions when the network is synchronous and there are no faults. Sailfish [21] addresses this issue by introducing a notion of timeout certificates to the protocol. Timeout certificates prevent situations where Byzantine replicas can cause correct replicas to skip voting. Timeout certificates additionally enable Sailfish to support an anchor in every round without pipelining. Sailfish also has a multi-leader variant, which pipelines multiple instances in a round.

Fast Commit Rule Revisited. Recall that the fast commit rule requires $2f + 1$ PROPOSE messages for vertices in round $r + 1$ that reference the anchor in round r . Suppose a Byzantine replica observes f Byzantine vertices that do not reference the anchor, and $f + 1$ correct vertices in round $r + 1$ that vote for the anchor in its DAG. This Byzantine replica can quickly propose a valid vertex in round $r + 2$ causing the f remaining correct replicas to jump to round $r + 2$. Since these f correct replicas will not vote in round $r + 1$, the normal Bullshark commit rule must be used since there are only $f + 1$ correct votes.

Timeout Certificates. The key problem is that a Byzantine replica can cause a subset of correct replicas to advance rounds too quickly and not vote. To solve this, Sailfish adds additional constraints to vertices so that Byzantine replicas cannot advance rounds too quickly. A valid vertex must now contain an edge to the anchor or a timeout certificate. A timeout certificate contains $2f + 1$ timeout messages, where a timeout message indicates that a replica did not receive the anchor after a certain amount of time. The previous attack fails with this change because if a Byzantine replica tries to propose a vertex without an edge to the anchor, then it must include a timeout certificate, which requires at least $f + 1$ correct replicas to timeout. Since correct replicas do not timeout in good-case conditions, no time-

out certificate can form and the fast commit rule will be satisfied at every correct replica.

This constraint on the vertices also enables an anchor in every round without pipelining. If an anchor in round $r + 1$ has an edge to the anchor in round r , then the extension invariant is satisfied. Otherwise, if the anchor in round $r + 1$ has a timeout certificate, then it is not possible for the anchor in round r to have committed, so it is safe for the anchor in round $r + 1$ to not reference the anchor in round r .

4.4 Discussion

Shoal, Shoal++, and Sailfish utilize the same certified DAG backbone so they inherit the same chain quality, fairness, data synchronization, and throughput properties.

Communication Complexity. The communication complexity of Shoal and Shoal++ remains at $O(n^3)$, since they do not add any extra messages. Sailfish also has the same complexity because timeout certificates can be reduced to $O(1)$ size using threshold signatures.

Latency. In terms of latency, Shoal still requires two DAG rounds or 6δ to commit anchors. However, with pipelining there is an anchor in every round, so non-anchor vertices only have a latency of 9δ . The queuing latency is on average 1.5δ since vertices are proposed every 3δ . The full consensus latency is thus 10.5δ .

Shoal++ reduces the commit latency of anchors to 4δ , and pipelines aggressively so that all vertices experience 4δ commit latency. By pipelining DAGs, Shoal++ also reduces the queuing latency down to 0.5δ for a total end-to-end latency of 4.5δ . Sailfish also uses the fast commit rule, so anchor commit latency is also 4δ . Non-anchor vertices require an extra DAG round to be included by an anchor, resulting in a commit latency of 7δ . However, applying the same pipelining optimizations to Sailfish results in a consensus latency of 4.5δ .

5 Uncertified DAG

The uncertified DAG family of protocols assume a DAG structure similar to a certified DAG. However, they **do not** require the DAG to satisfy the non-equivocation property. As a result, they eschew consistently broadcasting vertices and instead send vertices on a best-effort basis. Best-effort broadcast reduces latency by two

message delays compared to consistent broadcast but requires the ordering algorithm to have additional complexity to handle possibly inconsistent DAGs. We first present Cordial Miners [17], and then cover Mysticeti [4].

5.1 Cordial Miners

As with the certified DAG, the uncertified DAG is also divided into a number of integer-valued rounds. In each round, a source replica attempts to add a vertex to the DAG, where each vertex contains $n - f$ edges to vertices in the previous round. However, since replicas do not consistently broadcast vertices, it is possible for a Byzantine replica to equivocate and have multiple vertices in a given round.

DAG Structure. Cordial Miners logically groups every three rounds in the DAG into a *wave*. For example, rounds 0, 1, and 2 are in wave 1 rounds 3, 4, and 5 are in wave 2, etc. In the first round of a wave there is a pre-defined leader, and its associated vertex is the anchor. Note that if the leader is Byzantine, there may be multiple anchors due to equivocation. The second round in the wave is a voting round, where a vote is a vertex in round $r + 1$ which has an edge to an anchor in round r . The third round of the wave is the final voting round, where if enough votes are observed an anchor is committed.

Adding vertices to the DAG. When a replica wants to add a vertex to the DAG it sends a PROPOSE message containing this vertex to all replicas. Upon receiving a PROPOSE message with a new vertex, a replica adds the vertex to its local DAG **only** if it has received the entire causal history of the vertex. If replicas do not wait to have the entire causal history, then it is possible for a vertex to be committed without any correct replica having the entire causal history for that vertex. This violates the data availability property, which compromises liveness. If a replica is missing the causal history locally, it can ask the source replica to send over the missing vertices. Note that correct replicas may receive equivocating vertices. Although this is proof of Byzantine behavior, correct replicas still need to add these vertices to their DAGs because it is possible one of these vertices may be committed.

Advancing DAG rounds. As with certified DAGs, correct replicas must use timeouts in each DAG round to ensure liveness. Upon entering a new DAG round, replicas start a timer, and wait to receive at least $n - f$ vertices in the current round. Then, a replica checks whether the following conditions hold. If a replica is in the first round of a wave, it advances if it received an anchor or times out. If a replica is in the second round of a wave, it advances if it received a quorum of $n - f$ votes

for an anchor or times out. Finally, if a replica is in the third round of a wave, it advances if it commits an anchor or times out.

Totally Ordering the DAG. As with the certified DAG family of protocols, the ordering algorithm agrees on a hash chain of anchors. Once this sequence of anchors is committed, the causal history of each anchor can be committed in some deterministic order. To maintain safety, the commit rule must satisfy the extension invariant, which ensures that if an anchor is committed, then any future anchor in the hash chain must extend it.

Commit Rule. An anchor, A , in round r can be safely committed if there are $n - f$ votes in round $r + 2$ that have edge to a certificate for A . A certificate for A consists of $n - f$ votes in round $r + 1$ for A . The certificate ensures non-equivocation in that at most one anchor in a wave can receive $n - f$ votes (by quorum intersection). The commit rule satisfies the extension invariant because any vertex in round $r + 3$ must have an edge to one of the certificate votes in round $r + 2$ (by quorum intersection). Since any future anchor must be in round $r + 3$ or have a vertex in round $r + 3$ in its causal history, the extension invariant holds.

Final Total Order. Like in certified DAGs, given an anchor, the full hash chain of anchors is defined by recursively computing the parent anchor. The parent, A_p , of an anchor, A , is the latest anchor (highest round) in A 's causal history such that there is a vertex which references a certificate of A_p . A vertex which references a certificate is when there is a vertex in A 's causal history that has $n - f$ votes for A_p . Once an anchor is committed, the hash chain sequence of anchors is also committed. A replica will then iterate through each anchor in sequence order and execute the causal history of each anchor in some deterministic order.

5.2 Mysticeti

Mysticeti [4] uses the core Cordial Miners [17] protocol but adds aggressive pipelining similar to Shoal++ [3]. In each round, there are k proposer slots, which are totally ordered. A proposer slot runs an instance of a single-shot Cordial Miners. For a single-shot instance for a proposer replica i in round r , the first anchor is initialized to be a vertex from replica i in round r . Once a commit or skip decision is reached for this anchor, the single-shot instance terminates. Before a proposer slot is executed, a replica must wait for all prior proposer slots (in earlier rounds or smaller ranked slots in the same round) to either be committed or skipped. A replica then iterates through all committed slots in order and executes the causal history of each committed vertex in some deterministic order.

5.3 Discussion

A benefit of Cordial Miners and Mysticeti is that they require fewer signatures compared to certified DAG protocols. This is because they do not require forming certificates containing $n - f$ signatures for each vertex. A single signature is effectively shared as a certifying vote for several vertices.

Data Synchronization. A drawback of Cordial Miners and Mysticeti is that replicas must fetch the entire causal history of a vertex before adding it to the DAG. Otherwise, replicas may commit a vertex v , where no correct replica has some vertex, v' , in v 's causal history. This can happen if Byzantine replicas do not send their vertices to all replicas. Unlike with certified DAGs, this data synchronization step must occur on the critical path of consensus. In the worst case, correct replicas may have to synchronize on an arbitrary large amount of data, which can trigger timeouts even under benign network conditions. Furthermore, this synchronization step increases the communication complexity to $O(n^4)$, as each correct replica may have to send a causal history of $O(n)$ vertices of size $O(n)$ to $O(n)$ replicas.

Latency. For latency, Cordial Miners requires 3 DAG rounds to commit an anchor, where a DAG round has δ latency. Non-anchor vertices in the first round of a wave require 6 DAG rounds to commit since they must wait for the next anchor to be committed. Vertices in the second and third round of a wave require 5 and 4 DAG rounds respectively as they must also wait for the next anchor to be committed. In total, the commit latency is 5δ . Since vertices are proposed every δ , the expected consensus latency is 5.5δ .

Mysticeti aggressively pipelines so that each vertex has the same commit latency as the anchor latency (3δ). Unfortunately, even with good-case conditions it is not guaranteed that the k slots in a round will all be committed. This requires the $n - f$ replicas who voted for the first slot in a round to also vote for all the other slots. When this condition does hold, Mysticeti has a consensus latency of 3.5δ (3δ commit latency and 0.5δ queuing delay). Note, though, that not every message delay is created equal. Each message delay in Cordial Miners and Mysticeti requires sending a full data proposal which in practice can take longer to transmit than consensus metadata messages. In contrast, with certified DAGs only the first message of consistent broadcast requires sending the full data proposal.

6 Hybrid DAG

Hybrid DAG protocols combine a DAG-based data dissemination layer with a traditional leader-based BFT consensus protocol to get the best of both worlds. Instead of relying on a commit rule that counts the number of edges that reference a particular vertex, hybrid DAG protocols use a traditional BFT consensus protocol to commit vertices. A traditional BFT consensus protocol commits vertices with low latency, while the DAG layer ensures high throughput by parallelizing data dissemination across all replicas. We first describe the Autobahn [12] system, and then cover Star [8] and BBCA-Chain [19].

6.1 Autobahn

In Autobahn [12], replicas construct a DAG of data proposals. This DAG has a simpler structure compared to the other DAG protocols, since the DAG edges are not used for consensus. Periodically, the consensus protocol will commit snapshots of the DAG. Once a snapshot has been committed, replicas must fetch the data proposals of the snapshot using a data synchronization protocol. Finally, replicas can execute all the vertices in the snapshot in any common deterministic order. We first describe how the DAG layer works, followed by the consensus layer, and then finally the data synchronization protocol.

DAG Layer. Unlike prior DAG protocols, a valid vertex from a source replica i has only one edge to the previous vertex from replica i . The resulting DAG consists of n parallel hash chains, one per replica.

Autobahn uses *weak consistent broadcast* to disseminate vertices. Weak consistent broadcast follows the same protocol as consistent broadcast but instead of a certificate containing $n - f$ VOTE messages, it contains $f + 1$ VOTE messages. Certificates in weak consistent broadcast do **not** protect against equivocation. Instead, they prove that at least one correct replica has stored the corresponding data proposal for the vertex. This ensures that if a correct replica commits a vertex, the corresponding data proposal is guaranteed to be retrievable.

A replica can propose a new vertex as soon as its last vertex was delivered through weak consistent broadcast. Unlike prior DAG protocols, replicas do not proceed in synchronized rounds or wait for timeouts. Replicas propose vertices at their own pace as soon as their own vertices have been delivered. Vertices can be pipelined to further reduce latency.

Consensus Layer. Replicas may have diverging DAGs due to equivocation or asynchrony. The consensus layer reconciles these differences by ensuring that

replicas agree on common snapshots of the DAG. A snapshot consists of the latest vertex from each source replica (n certificates), representing the frontier of the DAG. To ensure that snapshots cover new vertices, correct replicas must propose snapshots that satisfy a coverage parameter. This coverage parameter indicates the number of hash chains that have at least one new vertex since the previous snapshot. It is typically set to $n - f$ (since at most f are faulty). The specific consensus protocol used is a slightly modified version of PBFT. However, in principle any consensus protocol can be used. Once a particular snapshot is committed, replicas fetch the causal history of all n vertices that represent the snapshot and commit them in some common deterministic order.

Data Synchronization. Autobahn takes advantage of the simpler DAG structure to have a faster data synchronization protocol. In Autobahn, a replica only votes for a vertex in weak consistent broadcast from a source replica once it has already voted for all prior vertices in that source replica's hash chain (causal history). Since replicas vote for vertices in order, if a vertex has $f + 1$ votes, then at least one correct replica from this set of $f + 1$ has stored the entire causal history. Thus, a replica can get the entire causal history of a vertex in just a single round-trip from asking the set of $f + 1$ replicas that voted for the vertex. This is in contrast to certified DAG protocols which must recursively synchronize, incurring multiple round-trips to fetch the entire causal history.

6.2 Star

Star [8] has a similar DAG layer and consensus layer, where the DAG layer consists of n parallel hash chains and the consensus layer uses a traditional BFT protocol like PBFT. The DAG layer also uses weak consistent broadcast to deliver vertices, but unlike Autobahn it does proceed in synchronized rounds. A replica can only proceed to the next round and propose a new vertex upon delivering $n - f$ vertices in the current round. The consensus layer runs a consensus instance per round, which agrees upon a set of at least $n - f$ vertices to output. Replicas can then order the committed vertices in some deterministic way. The DAG and consensus layers can also be pipelined to reduce latency.

6.3 BBCA-Chain

BBCA-Chain [20] uses an uncertified DAG but differs from Cordial Miners and Mysticeti in that it does not use a DAG-based commit rule for ordering. Instead, it leverages a traditional leader-based BFT protocol, BBCA, similar to PBFT, to

agree on a hash chain of leader vertices. BBCA messages can be piggybacked onto the messages to construct the DAG, so that it has minimal overhead.

Leader and non-leader vertices both have $n-f$ edges to vertices in the previous DAG round; however, they differ in how they are disseminated. Non-leader vertices are proposed in every DAG round using best-effort broadcast, while leader vertices are proposed every time BBCA outputs a valid ticket. This ticket is typically a commit certificate for the previous leader vertex, but can also be a set of timeout messages that indicate a quorum of replicas have not committed the previous leader vertex. The ticket is necessary so that the extension property is satisfied. If a leader vertex v is committed, then any future leader vertex must have a ticket which extends v .

6.4 Discussion

Communication Complexity. The simplified DAG structure reduces the communication complexity of the data layer to $O(n^2)$ for Autobahn and Star since each vertex only has a single edge instead of $n-f$. While consensus inputs are of size $O(n)$, using a traditional BFT consensus protocol still retains $O(n^2)$ communication complexity. Autobahn’s fast data synchronization may incur $O(n^3)$ in the worst case, as $O(n)$ replicas may need to synchronize with $O(n)$ replicas on $O(n)$ tips. BBCA-Chain still retains the uncertified DAG structure of Sailfish and Mysticeti, so its communication complexity is also $O(n^4)$.

Latency. Autobahn’s consensus latency is 4.5δ applying aggressive pipelining and an optimization for consensus to propose uncertified vertices. This includes δ for a vertex to be disseminated, 0.5δ to be included in a consensus proposal, and another 3δ for the consensus protocol to commit the vertex. Star’s consensus latency (including pipelining) includes 0.5δ for a request to be included in a vertex, 2δ for the vertex to be added to the DAG (weak consistent broadcast), and 3δ for consensus to terminate. For BBCA-Chain, vertices are proposed every δ , so on average the queuing delay is 0.5δ for a request to be included in a vertex. Leader vertices experience a latency of 3δ using BBCA. Non-leader vertices take δ to be disseminated by best-effort broadcast, 1.5δ to be included by the next leader vertex, and 3δ for BBCA to commit that vertex.

Chain Quality. Star achieves chain quality by ensuring that each consensus instance decides at least $n-f$ vertices, of which at least $n-2f$ are from correct replicas ($\geq \frac{1}{2}$ of total replicas). Autobahn does not cap the amount of vertices that can be committed from a particular consensus instance, so it is possible that a majority of committed vertices are from Byzantine vertices. Autobahn thus sacrifices

chain quality to provide each replica the same opportunity for progress.

Seamlessness. A core contribution of Autobahn is defining a property called seamlessness. Seamlessness is a property that attempts to capture the robustness of a partially synchronous protocol. In a truly robust system, a synchronous period should not be affected by asynchrony that occurred before. Specifically, seamlessness states that after a period of asynchrony there should not be a performance degradation that lasts once synchrony returns.

Traditional BFT protocols are not seamless since during asynchrony they may fail to make any progress causing a backlog of uncommitted requests to form. The following synchronous period must work off this entire backlog, causing latency to spike until this backlog is worked off completely. Certified DAG protocols are almost seamless, but not quite. During an asynchronous period, consensus may stall but vertices are continually added to the DAG. Once synchrony returns, and consensus commits a new vertex, these uncommitted vertices from the asynchronous period are also committed as part of the causal history. As a result, latency does not spike due to a large backlog. However, the data synchronization process to fetch the causal history of a committed vertex can take many round-trips, leading to a latency increase.

Uncertified DAGs (including BBCA-Chain) are not seamless, since data synchronization must occur on the critical path of consensus. After an asynchronous period, there could be a large amount of data to synchronize on, which can cause timeouts to fire, increasing latency. Star is also not seamless because during asynchrony a consensus instance is started per DAG round. Once synchrony returns, all these consensus instances need to terminate before new requests can be executed.

Autobahn is seamless since the DAG grows during asynchrony. Then, once a synchronous period returns the first consensus instance commits the entire sub-DAG that formed during asynchrony. The fast data synchronization protocol ensures that fetching the entire sub-DAG takes just a single round-trip.

7 Conclusion

This survey covered state-of-the-art results in partially synchronous DAG BFT. Significant progress has been made in reducing communication complexity and latency. However, recent results have not quite achieved optimal latency [2, 6]. Progress has also been made on practical issues such as improving fairness, censorship resistance, and data synchronization. One challenge in particular is achieving censorship resistance without committing duplicate transactions. Mir-BFT [25] proposed a solution, but further improvements would be interesting future work.

References

- [1] Sui Blockchain. (last accessed on 09/23/24). URL: <https://sui.io/>.
- [2] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, page 331–341, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465084.3467899.
- [3] Balaji Arun, Zekun Li, Florian Suri-Payer, Sourav Das, and Alexander Spiegelman. Shoal++: High throughput DAG BFT can be fast and robust! In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 813–826, Philadelphia, PA, April 2025. USENIX Association. URL: <https://www.usenix.org/conference/nsdi25/presentation/arun>.
- [4] Kushal Babel, Andrey Chursin, George Danezis, Lefteris Kokoris-Kogias, and Alberto Sonnino. Mysticeti: Low-latency dag consensus with fast commit path. *arXiv preprint arXiv:2310.14821*, 2023.
- [5] Leemon Baird. The swirls hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirls Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, 34:9–11, 2016.
- [6] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association.
- [7] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.
- [8] Sisi Duan, Haibin Zhang, Xiao Sui, Baohan Huang, Changchun Mu, Gang Di, and Xiaoyun Wang. Dashing and star: Byzantine fault tolerance with weak certificates. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 250–264, 2024.
- [9] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.
- [10] Adam Gagol, Damian Lesniak, Damian Straszak, and Michał Swietek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes, 2019. URL: <https://arxiv.org/abs/1908.05156>, arXiv:1908.05156.
- [11] Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, page 296–315, Berlin, Heidelberg, 2022. Springer-Verlag. doi:10.1007/978-3-031-18283-9_14.

- [12] Neil Giridharan, Florian Suri-Payer, Ittai Abraham, Lorenzo Alvisi, and Natacha Crooks. Autobahn: Seamless high speed bft. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, SOSP ’24, page 1–23, New York, NY, USA, 2024. Association for Computing Machinery. doi:[10.1145/3694715.3695942](https://doi.org/10.1145/3694715.3695942).
- [13] Neil Giridharan, Florian Suri-Payer, Matthew Ding, Heidi Howard, Ittai Abraham, and Natacha Crooks. Beegees: stayin’alive in chained bft. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 233–243, 2023.
- [14] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. Sbft: A scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 568–580, USA, 2019. IEEE. doi:[10.1109/DSN.2019.00063](https://doi.org/10.1109/DSN.2019.00063).
- [15] Mohammad M. Jalalzai, Jianyu Niu, and Chen Feng. Fast-hotstuff: A fast and resilient hotstuff protocol. *CoRR*, abs/2010.11454, 2020. URL: <https://arxiv.org/abs/2010.11454>, arXiv:2010.11454.
- [16] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC’21, page 165–175, New York, NY, USA, 2021. Association for Computing Machinery. doi:[10.1145/3465084.3467905](https://doi.org/10.1145/3465084.3467905).
- [17] Idit Keidar, Oded Naor, Ouri Poupko, and Ehud Shapiro. Cordial miners: Fast and efficient consensus for every eventuality. In *37th International Symposium on Distributed Computing (DISC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.
- [18] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine Fault Tolerance. *ACM Transactions on Computer Systems (TOCS)*, 27(4):7:1–7:39, January 2010. URL: <http://doi.acm.org/10.1145/1658357.1658358>, doi:[10.1145/1658357.1658358](https://doi.org/10.1145/1658357.1658358).
- [19] Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. Bbca-chain: Low latency, high throughput bft consensus on a dag. *Financial Cryptography and Data Security (FC’24)*, 2024.
- [20] Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. Bbca-chain: Low latency, high throughput bft consensus on a dag. In Jeremy Clark and Elaine Shi, editors, *Financial Cryptography and Data Security*, pages 51–73, Cham, 2025. Springer Nature Switzerland.
- [21] Nibesh Shrestha, Rohan Shrothrium, Aniket Kate, and Kartik Nayak. Sailfish: Towards Improving the Latency of DAG-based BFT . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1928–1946, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. URL: <https://doi.ieee.org/10.1109/SP61157.2025.00021>, doi:[10.1109/SP61157.2025.00021](https://doi.org/10.1109/SP61157.2025.00021).

- [22] Alexander Spiegelman, Balaji Aurn, Rati Gelashvili, and Zekun Li. Shoal: Improving dag-bft latency and robustness. *Financial Cryptography and Data Security (FC'24)*, 2024.
- [23] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2705–2718, New York, NY, USA, 2022. Association for Computing Machinery. doi:[10.1145/3548606.3559361](https://doi.org/10.1145/3548606.3559361).
- [24] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: The partially synchronous version. *arXiv preprint arXiv:2209.05633*, 2022.
- [25] Chrysoula Stathakopoulou, Tudor David, Matej Pavlovic, and Marko Vukolić. Mir-bft: High-throughput robust bft for decentralized networks. *arXiv preprint arXiv:1906.05552*, 2019.
- [26] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 347–356, New York, NY, USA, 2019. Association for Computing Machinery. doi:[10.1145/3293611.3331591](https://doi.org/10.1145/3293611.3331591).

The Bulletin of the EATCS

THE FORMAL LANGUAGE THEORY COLUMN

BY

GIOVANNI PIGHIZZINI

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

GENERALIZED DIRECTIONS ON A COMPASS

Tim A. Hartmann*

Abstract

Beside the four cardinal directions on a compass (North, East, South, West), there are more fine grained directions between them, such as NorthEast and EastNorthEast. They are formed by concatenating neighboring directions, but not arbitrarily: EastNorth and NorthEastEast are incorrect directions. We study the underlying (though not explicitly given) recursive naming procedure. As it turns out, these generalized directions on a compass are indeed unique and are formed by a tabled Lindenmayer system, usually used to describe cellular plant growth. As we show, deciding whether a given string actually forms a direction is solvable in linear time.

1 Introduction

A typical Western world compass rose contains, in clock-wise order, the four cardinal directions North(N), East(E), South(S), West(W).¹ All further directions are a combination of these four words. We consider the naming of those between N and E only, as all other other directions are formed symmetrically.

As the basic principle, the direction exactly half-way between, say N and E, is formed by fusing these two names together, in this case to NE but not to EN. The preference for NE over EN on this level is somewhat arbitrary, but the recursively formed names follow a pattern: The direction between N and NE is not ENN but is NNE. Similarly, the direction between E and NE is ENE, and not NEE. The underlying principle is to form, say NNE, is to put N before NE since N compared to NE has been derived in fewer steps. Hence also E and NE combine to ENE. Following this pattern, NE and NNE combine to the direction NENNE, and so on. (Section 2 will state the precise definition. See also Figure 1 and Figure 2.)

This is arguably the natural (recursive) *linguistic* interpretation of how directions on a compass are formed. In reality, though, this principle is soon abandoned in favor of a *numeric* measurement in degrees (0° for N, 90° for E and so on).

*CISPA Helmholtz Center for Information Security, Germany, tim.hartmann@cispa.de

¹Historically and internationally, there are other forms of compass roses, see [3, 9].

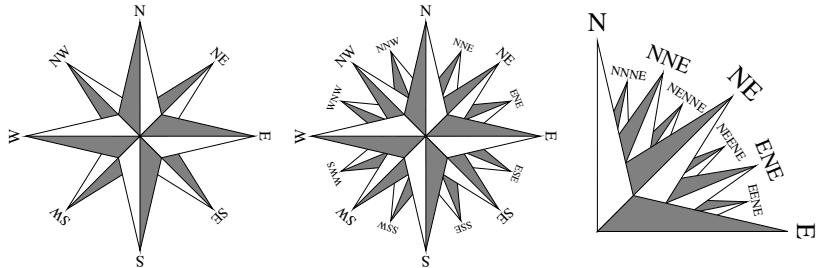


Figure 1: (Part of) compass roses, with ever more refined directions

Such a numeric measurement is easy to use, and, on the surface, this might be the reason to favor them over the seemingly complex linguistic interpretation. However, strictly algorithmically speaking, the linguistic interpretation is also easy to use. As we show, the recursive naming procedure produces unique names for the directions on the compass, and further, deciding whether a given string actually represents a direction on a compass (and if yes, which one) is decidable in linear time, i.e., deciding the following problem:

Input: A word $w \in \{N, E\}^*$ of length n .
 Question: Is w a direction on the compass strictly between N and E?

Further, there is a strong connection to rewriting systems. As we show, the generalized directions on a compass can be defined equivalently by a *context-free tabled Lindenmayer system* (tabled 0L-system). Here ‘context-free’ refers to the rules of the L-system. (In fact, the direction on the compass do *not* form a context-free language!) L-systems were introduced by Lindenmayer in 1968 to describe cellular plant growth [6]. The properties and expressiveness of different kinds of L-systems were extensively studied, see for example the book chapter of Kari et al. [5]. L-systems have been tied to natural languages before. Becerra-Bonache et al. [2] point to that certain aspects of natural language are not context-free and propose that L-systems pose a better model for natural languages.

Section 2 properly defines the directions on a compass. Section 3 introduces the theory then used in Section 4 to algorithmically detect directions. Finally, Section 5 points to the connection to rewriting systems.

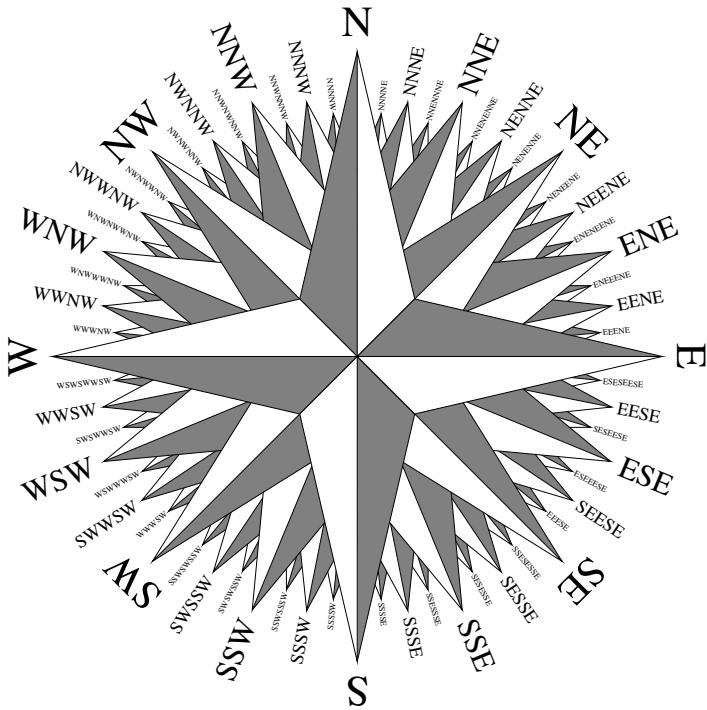


Figure 2: A detailed compass rose

2 Two Definitions

By symmetry, we may only consider the directions strictly between N and E. There is a *mechanical* and an *elegant* way to define the set of directions \mathcal{D} . The *mechanical* way defines \mathcal{D} as the union over $i = 1, 2, \dots$ of the leaves of the ordered rooted binary tree T_i , defined as follows:

- Tree T_1 only contains the root N with a right child E and NE as the left child of E. (That is, T_1 has in-order traversal N, NE, E and NE is the only leaf).
 - For $i \geq 1$, let L_i be the in-order traversal of the leaves of T_i . Then tree T_{i+1} results from T_i by adding for every leaf v with predecessor ℓ and successor r in L_i a left child ℓv and a right child rv . (For example, T_2 has in-order traversal N, NNE, NE, ENE, E and leaves NNE, ENE. See also Figure 3.)

The directions ℓv and rv obtained in above step 2. are located exactly half-way

between ℓ and v , respectively between r and v . Hence the in-order traversal is the order in which the directions appear clock-wise on the compass.

The *elegant* way defines \mathcal{D} by using the following set \mathcal{D}^- . Set \mathcal{D}^- contains each word of \mathcal{D} but with a hyphen at a suitable position. Instead of $NE \in \mathcal{D}$ we have $N-E \in \mathcal{D}^-$. Instead of a direction $\ell v \in \mathcal{D}$ and $rv \in \mathcal{D}$ formed in step 2 we have $\ell\text{-}v \in \mathcal{D}^-$ and $r\text{-}v \in \mathcal{D}^-$, respectively. For example $NNE \in \mathcal{D}$ becomes $N\text{-}NE \in \mathcal{D}^-$ (where the hyphen also indicates a pause while speaking).

Consider i such that tree T_i has leaf $uv \in \mathcal{D}$ (corresponding to $u\text{-}v \in \mathcal{D}^-$). Then v is the parent of uv and u is another ancestor of uv . More so, we know the two *children* of uv in the tree T_{i+1} . Indeed, the in-order traversal L_i contains the subsequence u, uv, v or its inversion. That is, we can define \mathcal{D}^- inductively by three simple rules:

1. $N\text{-}E \in \mathcal{D}^-$,
2. $f_N(u\text{-}v) \in \mathcal{D}^-$ if $u\text{-}v \in \mathcal{D}^-$ for some $u, v \in \{N, E\}^*$, and
3. $f_E(u\text{-}v) \in \mathcal{D}^-$ if $u\text{-}v \in \mathcal{D}^-$ for some $u, v \in \{N, E\}^*$; where

the mappings f_N, f_E are defined for inputs $u\text{-}v$ with $u, v \in \{N, E\}^*$ as follows:

$$\begin{aligned} f_N : u\text{-}v &\mapsto u\text{-}uv, \\ f_E : u\text{-}v &\mapsto v\text{-}uv. \end{aligned}$$

See Figure 3 for the derivation tree of $N\text{-}E$ using mappings f_N, f_E at most three times. Let ρ be the mapping that removes any symbol ‘-’ from its input word. Then \mathcal{D} is the set of words $\rho(w)$ with $w \in \mathcal{D}^-$. For example, $NNENENNE \in \mathcal{D}$, because it can be derived from $N\text{-}E$ by repeatedly applying f_N and f_E , and finally ρ :

$$N\text{-}E \xrightarrow{f_N} N\text{-}NE \xrightarrow{f_E} NE\text{-}NNE \xrightarrow{f_E} NNE\text{-}NENNE \xrightarrow{\rho} NNENENNE.$$

The exact position in the tree (and hence also the position on the compass) is revealed by the sequence in which f_N and f_E are applied. That is, whether a direction uvv (and similarly vuv) is the left or the right child of uv depends on whether the in-order traversal of T_i has subsequence u, uv, v or its inversion (where i is such that T_i has leaf uv). One can observe that whether its the inversion or not, flips with every use of f_E .

We will work with the elegant definition hereafter.

3 Detecting Directions

Essentially we are tasked to repeatedly *undo* the mappings f_N and f_E but without knowing the position of the hyphen. That is, we would like to compute *inverse*

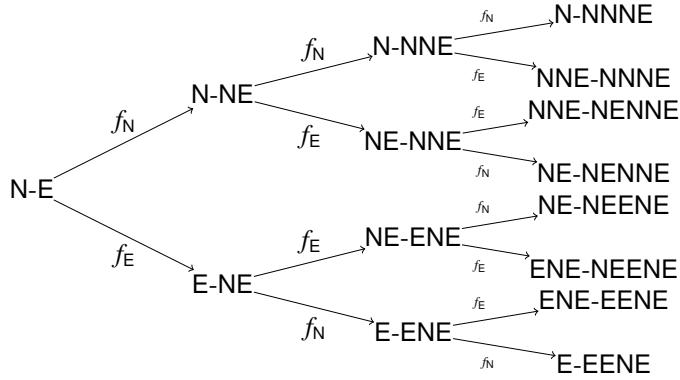


Figure 3: All derivations of N-E by applying f_N , f_E at most three times. Ignoring the hyphens and edge labels, this is also the subtree of T_4 rooted at NE.

mappings f_N^{-1} and f_E^{-1} , assuming they are unique. We do so by observing an alternative way to derive the directions on a compass. This alternative derivation turns out to be easy to undo.

We rely on functions h_N and h_E that modify the current word *locally*. More precisely, h_N and h_E are homomorphisms and their replacement for each type of character is defined as: For $X \in \{N, E\}$,

$$h_X : N \mapsto X, \quad h_X : E \mapsto NE, \quad h_X : - \mapsto -.$$

That is, the character ‘E’ is always replaced by ‘NE’, while ‘N’ is replaced by ‘N’ or ‘E’ depending on the subscript of h . Additionally, h_N and h_E simply ignore the separator ‘-’. Then, surprisingly, h_N and h_E provide an alternative way to derive the directions, for example NNE-NENNE:

$$N-E \xrightarrow{h_E} E-NE \xrightarrow{h_E} NE-ENE \xrightarrow{h_N} NNE-NENNE.$$

The sequence of the applied functions h_N and h_E resembles the *reverse* order in which f_N and f_E were applied. Hence we would like to decompose, for example NE-NENNE, as

$$NE-NENNE \xrightarrow{h_N^{-1}} NE-ENE \xrightarrow{h_E^{-1}} E-NE \xrightarrow{h_E^{-1}} N-E.$$

Then the order $h_N^{-1}h_E^{-1}h_E^{-1}$ unveils the original order f_E, f_E, f_N . This is no coincidence, as we will show.

3.1 The Inverses

It turns out that the inverses of h_N and h_E can be computed by parsing the given word w from left to right with very little lookahead. If w has prefix E , then it must be derived from a word with prefix N and using h_E . If it has prefix NN , it must be derived from a word with prefix N and using h_N . If it has prefix NE , it must be derived from a word with prefix E , and either h_N or h_E was used. The remaining part of w can be decomposed recursively. We thus can state explicit rules to compute h_N^{-1} and h_E^{-1} . For every suffix $v \in \{N, E, -\}^*$ and every $X \in \{N, E\}$,

1. $h_E^{-1}(Ev) = Nh_E^{-1}(v)$, and $h_N^{-1}(Ev)$ is not defined;
2. $h_N^{-1}(NNv) = Nh_N^{-1}(Nv)$, and $h_E^{-1}(NNv)$ is not defined;
3. $h_X^{-1}(NEv) = Eh_X^{-1}(v)$;
4. $h_N^{-1}(N) = N$, and $h_E^{-1}(N)$ is not defined;
5. $h_N^{-1}(N-v) = Nh_N^{-1}(-v)$, and $h_E^{-1}(N-v)$ is not defined;
6. $h_X^{-1}(-v) = -h_X^{-1}(v)$; and
7. $h_X(\epsilon) = \epsilon$.

We note that whether we consider directions with hyphen or not does not influence h_N^{-1}, h_E^{-1} . Recall that ρ is the mapping that removes any occurrence of ‘-’ from the input word.

Lemma 3.1. *Let $w = u-v \in \mathcal{D}^- \setminus \{N-E\}$ and $X \in \{N, E\}$. Then $\rho(h_X^{-1}(w)) = h_X^{-1}(\rho(w))$. Further, $h_X^{-1}(u-v) = h_X^{-1}(u)-h_X^{-1}(v)$ and $h_X^{-1}(uv) = h_X^{-1}(u)h_X^{-1}(v)$.*

Proof. For inputs without infix $N-E$, above rules 1.-7. essentially ignore the symbol ‘-’. That is $\rho(h_X^{-1}(w)) = h_X^{-1}(\rho(w))$ for every $X \in \{N, E\}$ and word $w \in \{N, E, -\}^*$ without infix $N-E$. In fact, this observation applies to $w \in \mathcal{D}^- \setminus \{N-E\}$. That is, we claim that no word $w \in \mathcal{D}^- \setminus \{N-E\}$ has infix $N-E$. Recall that w has form $\ell-v$ or $r-v$ where ℓ, v, r is a subsequence in the in-order traversal L_i of the tree T_i with leaf v . Hence ℓ or r has suffix N . However, inductively we observe that no word in L_i but N itself has suffix N . Indeed, in L_1 only N itself has suffix N . By induction on i , in L_i again only N has suffix N since all new direction but $N-N^{i-1}E$ are formed by a concatenation words with suffix E .

Thus $\rho(h_X^{-1}(w)) = h_X^{-1}(\rho(w))$ for $w = u-v \in \mathcal{D}^- \setminus \{N-E\}$ and $X \in \{N, E\}$. Particularly, the rules of 1.-7. are applied to u and v independently, such that $h_X^{-1}(u-v) = h_X^{-1}(u)-h_X^{-1}(v)$. Finally, we observe that $h_X^{-1}(uv) = h_X^{-1}(\rho(u-v)) = \rho(h_X^{-1}(u-v)) = \rho(h_X^{-1}(u)-h_X^{-1}(v)) = h_X^{-1}(u)h_X^{-1}(v)$. \square

Lemma 3.2. *For $w \in \mathcal{D} \setminus \{\text{NE}\}$, at most one of h_N^{-1} and h_E^{-1} is defined.*

Proof. For the directions in L_2 , NNE and ENE, only h_N^{-1} and only h_E^{-1} , respectively, is defined. By induction, any further word derived from ENE (in terms of f_N and f_E) contains the infix EE, hence only h_E^{-1} is defined. Similarly, any further word derived from NNE (in terms of f_N and f_E) contains the infix NN, hence only h_N^{-1} is defined. \square

3.2 Uniqueness of the Directions

For N-E, the mapping h_N^{-1} exactly undoes f_N , and h_E^{-1} exactly undoes f_E .

Observation 3.3. *For $X, Y \in \{N, E\}$, we have $h_X^{-1}(f_Y(N-E)) = N-E$ if $X = Y$, and $h_X^{-1}(f_Y(N-E))$ is undefined if $X \neq Y$.*

Proof. We have $h_X^{-1}(f_X(N-E)) = h_X^{-1}(X-NE) = N-E$ for $X \in \{N, E\}$, but $h_N^{-1}(f_E(N-E)) = h_N^{-1}(E-NE)$ nor $h_E^{-1}(f_N(N-E)) = h_E^{-1}(N-NE)$ is defined. \square

Every direction in \mathcal{D}^- other than N-E is derived from N-E. The key insight is that h_N^{-1} and h_E^{-1} commute with f_N and f_E .

Observation 3.4. *Let $X, Y \in \{N, E\}$ and $w \in \mathcal{D}^- \setminus \{N-E\}$. Then $h_Y^{-1}(f_X(w)) = f_X(h_Y^{-1}(w))$.*

Proof. A word $w \in \mathcal{D}^-$ has format u_N-u_E for some words $u_N, u_E \in \mathcal{D}$. Then

$$\begin{aligned} h_Y^{-1}(f_X(u_N-u_E)) &= h_Y^{-1}(u_X-u_Nu_E) \\ &\stackrel{L.3.1}{=} h_Y^{-1}(u_X)-h_Y^{-1}(u_Nu_E) \\ &\stackrel{L.3.1}{=} h_Y^{-1}(u_X)-h_Y^{-1}(u_N)h_Y^{-1}(u_E) \\ &= f_X(h_Y^{-1}(u_N)-h_Y^{-1}(u_E)) \\ &\stackrel{L.3.1}{=} f_X(h_Y^{-1}(u_N-u_E)). \end{aligned}$$

\square

Now we have all the ingredients to detect directions on the compass by means of h_N and h_E . In general, a direction w is derived from N-E by consecutively applying n mappings $f_{Y_1}, \dots, f_{Y_n}, \rho$ for some integer $n \geq 0$ and subscripts $Y_1, \dots, Y_n \in \{N, E\}$. In other words, $w = (\rho \circ f_{Y_n} \circ \dots \circ f_{Y_1})(N-E)$. If we apply $h_{X_1}^{-1}$ to w , for some $X_1 \in \{N, E\}$, we can shift $h_{X_1}^{-1}$ in the sequence of applied $\rho \circ f_{Y_n} \circ \dots \circ f_{Y_1}$ just before f_{Y_1} , using Lemma 3.1 and Observation 3.4. Then, by Observation 3.3, $h_{Y_1}^{-1}$ and f_{Y_1} cancel out if and only if $X_1 = Y_1$. In other words,

$$(h_{Y_1}^{-1} \circ \rho \circ f_{Y_n} \circ \dots \circ f_{Y_2} \circ f_{Y_1})(N-E) = (\rho \circ f_{Y_n} \circ \dots \circ f_{Y_2})(N-E). \quad (1)$$

If $X_1 \neq Y_1$, then the term ends up not being defined. By applying this transformation iteratively, we obtain the following.

Theorem 3.5. *Let $X_1, \dots, X_n, Y_1, \dots, Y_n \in \{N, E\}$ for a positive integer n . Then $(h_{X_n}^{-1} \circ \dots \circ h_{X_2}^{-1} \circ h_{X_1}^{-1} \circ \rho \circ f_{Y_n} \circ \dots \circ f_{Y_2} \circ f_{Y_1})(N-E)$ equals NE if $(X_1, \dots, X_n) = (Y_1, \dots, Y_n)$, and is not defined otherwise.*

Proof. by Equation (1), $(h_{X_n}^{-1} \circ \dots \circ h_{X_2}^{-1} \circ h_{X_1}^{-1} \circ \rho \circ f_{Y_n} \circ \dots \circ f_{Y_2} \circ f_{Y_1})(N-E)$ is only defined for $X_1 = Y_1$, and is equal to $(h_{X_n}^{-1} \circ \dots \circ h_{X_2}^{-1} \circ \rho \circ f_{Y_n} \circ \dots \circ f_{Y_2})(N-E)$. By repeating this argument for $h_{X_2}^{-1}, h_{X_3}^{-1}, \dots, h_{X_n}^{-1}$, the statement follows. \square

Hence, given that we can consecutively apply $h_{X_1}^{-1}, h_{X_2}^{-1}, \dots, h_{X_n}^{-1}$ to some word w and eventually obtain NE, certifies that $w \in \mathcal{D}$ and that f_{X_1}, \dots, f_{X_n} are the applied rules. Vice versa, any direction $(\rho \circ f_{Y_n} \circ \dots \circ f_{Y_2} \circ f_{Y_1})(N-E) \in \mathcal{D}$ can be transformed to NE by consecutively applying $h_{Y_1}^{-1}, h_{Y_2}^{-1}, \dots, h_{Y_n}^{-1}$. By Lemma 3.2, we then have that exactly one of h_N^{-1} and h_E^{-1} is applicable to any direction $\mathcal{D} \setminus \{\text{NE}\}$, and hence each direction has its unique derivation in terms of the sequence of f_E and f_N . In other words, each direction \mathcal{D} defines a unique direction on the compass.

4 Algorithms

First, we observe a *simple* but slow algorithm, then we develop a *fast* algorithm. The findings of the last section immediately imply the *simple* algorithm: If the input is NE, we accept it as a direction. Otherwise, if none or both of h_N^{-1} and h_E^{-1} are applicable to the input, we may reject the input as not being a direction. Otherwise, we apply the unique mapping that is applicable to the input and recurse. Whichever mapping was applicable, reveals the derivation from the initial direction NE. For an input w of length n , this approach parses about n times a word of length at most length n , which summarizes to a run time of $O(n^2)$. This analysis is tight as shown by the example $N^k E$ for an integer k , where, with every recursion, the input length shrinks only by one.

Now we develop the *fast* algorithm. Sometimes the length of a word shrinks considerably when h_N^{-1} or h_E^{-1} is applied, such that we can hope for fewer than n recursions. We consider three cases.

Case Infix $N^k E$, $k \geq 2$ Consider the extreme case of $N^k E$ for some integer $k \geq 2$. Then $k - 1$ steps are needed to decompose the input to NE:

$$N^k E \xrightarrow{h_N^{-1}} N^{k-1} E \xrightarrow{h_N^{-1}} \dots \xrightarrow{h_N^{-1}} NE$$

$k-1$ applications of h_N^{-1}

The situation is similar if the input w has infix $N^k E$. In this case, the rules for computing h_N^{-1} and h_E^{-1} imply that w can only be decomposed by $h_{Y_k}^{-1} \circ \dots \circ h_{Y_1}^{-1}$ if $(Y_k, \dots, Y_1) = (N, \dots, N)$.

Observation 4.1. Let $u, v \in \{N, E\}^*$ and integer $k \geq 2$. Then $(h_{Y_{k-1}}^{-1} \circ \dots \circ h_{Y_1}^{-1})(uN^kEv)$ equals $(h_{Y_{k-1}}^{-1} \circ \dots \circ h_{Y_1}^{-1})(u)NE(h_{Y_{k-1}}^{-1} \circ \dots \circ h_{Y_1}^{-1})(v)$ if $(Y_{k-1}, \dots, Y_1) = (N, \dots, N)$, and is not defined otherwise.

Thus, if the input w has infix $N^k E$ for some $k \geq 2$, then w may only consist of blocks of form $N^{k'} E$ where $k' \geq k - 1$. Indeed, otherwise w has prefix $N^\ell E$ or infix $EN^\ell E$ for some integer $\ell < k - 1$. That means that w is derived from a word w' that has prefix E or has infix EE . By induction, one can show that w' must be derived from ENE . However, any word with an infix NN , such as w , must be derived from NNE , contradicting that w is derived from w' .

These findings justify the following subroutine. Parse the input w from left to right to determine the largest integer k such that w contains infix $N^k E$. This can be done in time linear in n , the length of w . We know that a direction w has form $N^{k_1} E N^{k_2} E \dots N^{k_m} E$ for some integers $k_1, k_2, \dots, k_m \in \{k, k-1\}$ and $m \geq 1$. If that is not the case, we can safely reject the input w . Otherwise, we know that w decomposes to $(h_N^{-1})^{k-1}(w) = u_1 \dots u_m$ where $u_i = E$ for $k_i = k-1$ and where $u_i = NE$ for $k_i = k$. An example with $k = 3$ is

$$NNNENNENNNE = N^3 EN^2 EN^3 E \xrightarrow{(h_N^{-1})^2} NEENE.$$

Hence, by a second parse of w , we can determine the integers $k_1, \dots, k_m \in \{k, k-1\}$ and then directly state $(h_N^{-1})^{k-1}(w)$, hence perform $k-1$ decomposition steps at once. The run time is linear in n . Let us perform this step whenever $k \geq 2$. Then every part u_1, \dots, u_m is shortened to a part v_1, \dots, v_m that shrunk by a factor of at least $2/3$. A border case is NNE that is only shrunk to NE .

Case Infix $E^k NE$, $k \geq 2$ We have a similar situation if the input word w contains a long sequence of E 's. Such a maximum length infix E^k with $k \geq 2$ must be followed by NE , as h_E must have been the last derivation step. Consider that w has form uE^kNEv for some $u, v \in \{N, E\}^*$. Then $h_E^{-1}(uE^kNEv) = h_E^{-1}(u)N^kEh_E^{-1}(v)$. Now we are in the same case as discussed before, which is that the input has an infix $N^k E$. Hence, our algorithm may determine the largest k such that the input has infix $E^k NE$. We decompose once via h_E^{-1} and then proceed as we did for the case infix $N^k E$, $k \geq 2$. Then again the input shrinks by a factor of at least $2/3$. Similarly to before, this step is possible in time $O(n)$.

Remaining Case In the remaining case, the input w neither has infix N^k nor infix E^k , for every $k \geq 2$. Then w has form $u_1 \dots u_m$ where $u_i \in \{NE, ENE\}$ for every $i \in \{1, \dots, m\}$. That is, w decomposes to $h_E^{-1}(u) = v_1 \dots v_m$ where $v_i = E$ if $u_i = NE$, and where $v_i = NE$ if $u_i = ENE$; for $i \in \{1, \dots, m\}$. Similarly to before, our algorithm can perform this step in time linear in the input length. Crucially, we again shorten the word length by at least a factor of $2/3$, since every part is shortened by at least this factor. For example,

$$NE\ NE\ ENE \xrightarrow{h_E^{-1}} h_E^{-1}(NE)h_E^{-1}(NE)h_E^{-1}(ENE) = EENE.$$

Run Time To summarize, three cases of the currently considered word w are possible. Either there is a long sequence of N 's, a long sequence of E 's, or w has a simple format. In every case, we can perform one or more decomposition steps at once in time linear in the length of the current word. The result is that recursively we shrink the input length by factor of $2/3$. Therefore, the recurrence relation for the run-time depending on the initial input length n is $T(n) = T(\frac{2}{3}n)O(n)$. This recurrence relation is solved by the third case of the master theorem, as for example presented in the text book by Cormen et al. [4]. (That is $T(n)$ has form $T(n) = a \cdot T(\frac{n}{b}) + f(n)$ with $f(n) \in O(n)$ with $a = 1$ and $b = \frac{3}{2}$. We have that $f(n) \in \Omega(n^{\log_{3/2} a + \varepsilon})$ for $\varepsilon = 0.5 > 0$ and $af(\frac{n}{b}) < c \cdot f(n)$ for $c = \frac{4}{3} < 1$ for all sufficiently large n .) Hence we conclude that the overall run time is $T(n) = O(n)$.

5 Connection to Rewriting Systems

The key step to understand the directions \mathcal{D} on a compass was rephrasing them in terms of a rewriting system. As a consequence of Theorem 3.5, that is:

- $NE \in \mathcal{D}$, and
- $h_X(u) \in \mathcal{D}$ if $u \in \mathcal{D}$ and $X \in \{N, E\}$.

Hence \mathcal{D} is defined by a *context-free tabled Lindenmayer system* (0L-system) on the alphabet $\{N, E\}$, start-word NE and mappings $\{h_N, h_E\}$. Here *context-free* refers to that the production rules defined by h_N, h_E are context-free and does not imply that \mathcal{D} is context-free. Further, *tabled* refers two that there is more then one way to rewrite a word at any given step, here either by h_N or by h_E . L-systems were introduced by Lindenmayer in 1968 to describe growth of multicellular organisms [6]. We point to the monograph on this matter of Prusinkiewicz & Lindenmayer [7]. The properties and expressiveness of different kinds of L-systems were extensively studied, see for example the book chapter of Kari et al. [5]. As a tabled

0L-system, \mathcal{D} is a context-sensitive language [8]. On the other hand, we can observe that \mathcal{D} is not a context-free language by considering the pumping lemma [1] on the word $N^kEN^kENN^kE \in \mathcal{D}$ for an integer k . Nevertheless, as we saw, the directions \mathcal{D} can be recognized in linear time.

6 Acknowledgments

I thank Gerhard Woeginger and Jakob Greilhuber for helpful discussions.

References

- [1] Yehoshua Bar-Hillel, Micha Perles, and Eliahu Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14(1-4):143–172, 1961.
- [2] Leonor Becerra-Bonache, Suna Bensch, and María Dolores Jiménez-López. The linguistic relevance of lindenmayer systems. In Joaquim Filipe, Ana L. N. Fred, and Bernadette Sharp, editors, *ICAART 2010 - Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 2 - Agents, Valencia, Spain, January 22-24, 2010*, pages 395–402. INSTICC Press, 2010.
- [3] Qun Rene Chen. Cardinal directions in chinese language: their cultural, social and symbolic meanings. *ETC: A Review of General Semantics*, 66(2):225–239, 2009.
- [4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- [5] Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. L systems. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, pages 253–328. Springer, 1997.
- [6] Aristid Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology*, 18(3):280–315, 1968.
- [7] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. The virtual laboratory. Springer, 1990.
- [8] Grzegorz Rozenberg. TOL systems and languages. *Inf. Control.*, 23(4):357–381, 1973.
- [9] Léopold de Saussure. L’origine de la rose des vents et l’invention de la boussole. *Archives des sciences physiques et naturelles*, 5:149–181, 1923.

The Bulletin of the EATCS

The Bulletin of the EATCS

THE EDUCATION COLUMN

BY

DENNIS KOMM AND THOMAS ZEUME

ETH Zurich, Switzerland and Ruhr University Bochum, Germany
dennis.komm@inf.ethz.ch and thomas.zeume@rub.de

ASSESSING GPT PERFORMANCE IN A PROOF-BASED UNIVERSITY-LEVEL COURSE UNDER BLIND GRADING

Ming Ding

Department of Computer Science, ETH Zurich

ming.ding@inf.ethz.ch

Federico Soldà

Department of Computer Science, ETH Zurich

federico.solda@inf.ethz.ch

Weixuan Yuan

Department of Computer Science, ETH Zurich

weyuan@inf.ethz.ch

Rasmus Kyng

Department of Computer Science, ETH Zurich

kyng@inf.ethz.ch

Abstract

As large language models (LLMs) advance, their role in higher education, particularly in free-response problem-solving, requires careful examination. This study assesses the performance of GPT-4o and o1-preview under realistic educational conditions in an undergraduate algorithms course. Anonymous GPT-generated solutions to take-home exams were graded by teaching assistants unaware of their origin. Our analysis examines both coarse-grained performance (scores) and fine-grained reasoning quality (error patterns). Results show that GPT-4o consistently struggles, failing to reach the passing threshold, while o1-preview performs significantly better, surpassing the passing score and even exceeding the student median in certain exercises. However, both models exhibit issues with unjustified claims and misleading arguments. These findings highlight the need for robust assessment strategies and AI-aware grading policies in education.

1 Introduction

The emergence of large language models (LLMs) has revolutionized how we interact with artificial intelligence (AI), and garnered significant attention from researchers and the public. While LLMs have demonstrated impressive capabilities, their true level of intelligence and reasoning remains a subject of debate.

The classical Turing Test proposes that a machine demonstrating human-like responses in conversation could be considered intelligent. Over the past few years, substantial efforts have been devoted to evaluating LLMs from various angles [3]. For example, LLMs can generate essays with their quality rated higher than those produced by humans [14]; pass questions involving communication skills, ethics, empathy, and professionalism in a United States Medical Licensing Examination (USMLE) [2]; achieve passing scores on the reading comprehension test of the Program for International Student Assessment (PISA), a global standardized student assessment [31]; and demonstrate strong performance in solving middle school-level math word problems, with multiple LLMs achieving passing scores and some exceeding 90% accuracy [32]. However, existing evaluation protocols may fall short of comprehensively assessing their reasoning and problem-solving capabilities. Most benchmarks involve structured datasets, such as multiple-choice or numeric questions, which may not fully capture the complexities of real-world tasks. While prior research has demonstrated their potential to assist humans or improve responses through prompt engineering [28, 33], there remains a need to evaluate LLMs in scenarios that require independent reasoning and creativity.

The study presented in this article addresses this gap by evaluating GPT models' performance in solving complex free-response questions, specifically proof-based problems from an undergraduate algorithms course. The tasks require not only creative algorithm design but also rigorous justification of correctness—a skill fundamental to human cognitive problem-solving. To ensure unbiased evaluation, we anonymously submitted GPT-generated responses to exam graders, who evaluated them under the assumption that they were written by students. Unlike prior work that focuses on coarse-grained metrics such as pass/fail rates or overall accuracy, we conduct a fine-grained analysis by examining grader feedback and error patterns. This provides deeper insights into LLMs' limitations in proof-based and free-response problem-solving. Beyond assessing performance, our study contributes to the broader discussion on AI's role in education by highlighting both its potential and its constraints. By exploring these aspects, we aim to clarify how AI can reshape learning and assessment practices, as well as the thresholds it must surpass to genuinely emulate undergraduate students.

Recent advancements in LLMs have introduced several competing products, such as OpenAI's ChatGPT [22], Google's Gemini [11], GitHub's Copilot [10], and SciSpace [27]. As these models continue to improve, their progress necessitates

a reevaluation of the role and effectiveness of academic assessments in the era of AI-assisted learning. In this work, we focus on ChatGPT-4o and o1-preview, the latest models from OpenAI available during our experiment period. GPT-4o (“o” for “omni”) is a multilingual, multimodal generative pre-trained transformer developed by OpenAI and released in May 2024. In September 2024, OpenAI released o1-preview, the first of a new series of products, with the full version released later. Despite their rapid development, the performance of OpenAI’s latest models in realistic educational settings remains largely unexamined. Our study evaluates their effectiveness in solving complex academic problems.

With a particular focus on GPT-4o and o1-preview, our study makes the following key contributions:

1. Our study is conducted in real educational settings with authentic grading conditions, ensuring that the evaluation closely reflects actual academic environments. The experimental design serves as a reference for measuring AI performance in an unbiased and reliable manner. Specifically, we implement *blind grading*, ensuring that the teaching team remains unaware of the experiment and the origin of the submissions. This prevents any potential bias in evaluation. Additionally, we generate solutions using *naive prompts*—without providing course materials or additional guidance—ensuring that GPT does not receive any advantages from instructor input. Different from other benchmarks that are designed to explore the upper limits of LLM problem-solving capabilities, our approach aims to establish a realistic *lower bound* on model effectiveness in educational environments, reflecting how LLMs might perform when used by students in typical, unstructured settings.
2. We use a set of original, highly challenging exercises drawn from take-home exams in an undergraduate algorithms theory course. These problems are designed to be difficult even with full access to course materials, ensuring that the evaluation assesses genuine problem-solving skills rather than mere retrieval-based responses. Our selected exercises are original, which sets us apart from existing popular evaluations on standardized examinations, which often have extensive learning material available online for trainees (including large sets of example questions and answers).

Moreover, our dataset consists of free-response, proof-based questions that demand multi-stage reasoning. While AI’s abilities have been extensively studied in structured tasks like coding [34] and multiple-choice questions [24], there is limited investigation into AI’s performance on complex and proof-based problems. Our dataset therefore provides a strong benchmark

for evaluating GPT’s capacity to produce correct, logically consistent, and well-structured responses to expert-level questions.

In Section 6, we present a selection of exercises along with their AI-generated solutions and the corresponding grader feedback and comments. The complete dataset is made available in the appendix of the full version of this paper [7].

Our work not only complements existing benchmarks focused on closed-ended or structured problems but also enriches the landscape of high-difficulty evaluations, such as BIG-Bench Hard [30], by providing open-ended, proof-based tasks.

3. We employ expert human grading to ensure a precise and trustworthy evaluation of both student and AI-generated solutions. Although this approach requires substantial human effort, it significantly enhances the reliability of comparisons across students, GPT-4o, and o1-preview. Additionally, our graders provide detailed feedback and comments, allowing for a *fine-grained analysis* of AI errors. By categorizing error patterns into unjustified claims, misleading statements, and mathematical inaccuracies, this qualitative assessment goes beyond numerical scoring, and provides insights into how the models function at a deeper level and highlights specific areas where improvements are needed. This depth of analysis distinguishes our study and offers clearer insights into AI’s strengths and limitations in academic problem-solving.

2 Related Work

Large Language Models (LLMs), such as GPT-4, have been extensively evaluated in mathematics and computer science courses to assess their reasoning and problem-solving capabilities. The evaluation of LLMs has been extensively reviewed [3], we focus here on the studies most relevant to our work.

Tests against standard datasets. A comprehensive description of how LLMs are traditionally evaluated can be found, for instance, in the GPT4 Technical Report [21]. In mathematics, the GSM8K dataset [6] contains 8.5K high-quality, linguistically diverse grade-school math word problems designed to evaluate and enhance the mathematical reasoning abilities of language models. Hendrycks et al. proposed the more challenging MATH dataset [13], which includes 12.5K problems spanning a wide range of topics and difficulty levels. Recent LLMs have also been evaluated on a variety of benchmarks: the AIME 2024 benchmark,

which features advanced competition-level mathematics problems [18]; the GPQA Diamond dataset [24], which focuses on numerical and multiple-choice questions; and the MMLU benchmark [12], a suite of multiple-choice questions covering 57 subjects. Additionally, the MathArena platform¹ offers a live leaderboard and curated mathematical challenges for benchmarking LLMs. A recent study by MathArena researchers [23] evaluated LLMs on the 2025 USAMO and found that, despite partial success by Gemini-2.5-Pro, all models struggled with proof generation. Other evaluations include ARC, the AI2 Reasoning Challenge [5]; WinoGrande, an adversarial Winograd schema challenge at scale [26]; HumanEval, which tests functional correctness in code generation from docstrings [4]; DROP, a reading comprehension benchmark requiring discrete reasoning over paragraphs [8]; and HellaSwag [36]; and others. Most recently, Latif et al. [17] evaluated the ChatGPT o1-preview model on established benchmarks. They observed that while o1-preview excelled in structured tasks, it displayed limitations in problem-solving and adaptive reasoning. Notably, it outperformed undergraduate and postgraduate participants in critical thinking, systematic thinking, data literacy, and creative thinking tasks, and surpassed the highest human scores in scientific reasoning.

The use of LLMs in higher education. The application of AI in higher education has also been widely explored in recent years. For example, Nikolić et al. [20] conducted a multi-institutional, multi-disciplinary evaluation of various Generative AI tools, including ChatGPT-3.5, ChatGPT-4, Copilot, Gemini, SciSpace, and Wolfram, across ten engineering subjects in seven Australian universities. The study evaluated AI performance on diverse assessment types. The tasks were rated as Pass, Fail, Component Pass, or Possible Pass, providing a coarse-grained evaluation of AI's educational capabilities. In addition, Susnjak et al. [29] explored how LLMs challenge the integrity of online exams. Their work emphasized multi-step reasoning strategies, guiding LLMs through iterative self-review and refinement to improve their responses. Yeadon et al. [34] evaluated the performance of GPT-3.5 and GPT-4 in university-level physics coding assignments using Python, concluding that AI-generated work closely approaches the quality of university students' work while remaining detectable by human evaluators.

Tests with human comparison. Several studies have directly compared AI and human performance in academic settings. Stribling et al. [28] assess the capability of GPT-4 to answer questions from nine graduate-level final examinations for scientific trainees in the biomedical sciences. They also examine the impact of multiple styles of GPT-4 queries (prompt patterns) and compare results to student performance. Grading has been performed blinded for most examinations.

¹<https://matharena.ai>

Kipp [16] tested GPT-4o on 30 unique anatomical multiple-choice questions from German medical licensing exams, where it consistently outperformed medical students across six state exams. In the study of Bayer et al. [1], ChatGPT-4, GPT-4o, and Google Gemini were tested on a physiology exam for veterinary students, with their results compared to those of the students. ChatGPT-4o achieved the highest score with 90% accuracy (18/20 correct answers), followed by Google Gemini at 75% (15/20), and ChatGPT-4 at 60% (12/20), all outperforming the students' average accuracy of 70% (13.99/20). Finally, Richards et al. [25] conducted a dual-anonymous “quality assurance” marking exercise on four end-of-module assessments within a distance-learning computer science curriculum. A total of 90 scripts, either student-submitted or ChatGPT-generated, were blindly graded. All ChatGPT-generated undergraduate scripts received at least a passing grade, whereas none of the ChatGPT-generated postgraduate scripts achieved a passing grade.

High-difficulty datasets. Several recent studies have explored the limitations and capabilities of large language models (LLMs) in proof-based and symbolic reasoning tasks using relatively small, high-difficulty datasets. BIG-Bench Hard [30] focuses on a suite of 23 challenging tasks from BIG-Bench that were found to be beyond the capabilities of current language models. These tasks are those where prior language model evaluations did not outperform the average human rater, making them valuable for stress-testing reasoning abilities. Building on this, Program-Aided Language models (PAL) [9] were evaluated on 13 tasks spanning mathematical, symbolic, and algorithmic reasoning drawn from BIG-Bench Hard and other sources. DeepMind’s AlphaProof system² evaluated model performance on six problems from the International Mathematical Olympiad (IMO), four of which were successfully solved. Despite the small dataset, this evaluation demonstrated that carefully selected high-difficulty problems can yield meaningful insights into model capabilities—enough to support claims of “silver medal” performance. Similarly, the miniF2F benchmark [35] for the Lean theorem prover contains a test set of 20 formalized IMO problems, with only six problems having fully verified Lean proofs. This benchmark highlights the difficulty of formal reasoning tasks and has become a standard for evaluating autoformalization and proof synthesis systems in low-data, high-complexity regimes.

3 Methodology

3.1 Experiment Design

The study presented in this article aims to assess the effectiveness of LLMs in performing algorithm exercises assigned to university students, using a blinded grading approach to evaluate solutions written by both students and AI.

²<https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>

Course and examination information. The study was conducted in a third-year undergraduate-level course at ETH Zurich in the Computer Science Department.³ As a core course in the Bachelor’s major “Theoretical Computer Science”, the class tends to attract a more specialized group of students, which introduces some selection bias compared to a typical undergraduate cohort. This introductory course on algorithm theory carries 8 ECTS and involves an estimated workload of 200 hours over a 14-week semester. Topics covered include Randomized Search Trees, Point Location, Bootstrapping Algorithms, Linear Programming, Randomized Algebraic Algorithms, and Parallel Algorithms.

The course evaluation comprises two take-home exams featuring free-response proof-based questions.⁴ For each take-home exam, students have two weeks to submit solutions, with unrestricted access to all course materials. They are required to propose algorithmic solutions and provide formal proof of correctness and complexity analysis of proposed algorithms. All solutions must be typeset in L^AT_EX and submitted as PDF.

Since this is a take-home task, students have access to the Internet and potentially AI tools. To ensure academic integrity as much as possible, besides requiring students to cite any external sources for results not covered in the lectures, the course instructors take additional measures in exercise design. Before release, each exercise undergoes a rigorous design process that includes: (1) Non-expert validation – Ensuring that no off-the-shelf solutions are readily available online or in past problem sets. (2) Expert validation – Conducting test-solving sessions among the teaching team to verify the problem’s clarity, difficulty, and originality. These measures help maintain exercises’ effectiveness while minimizing the risk of students relying on existing solutions.

Data collection and grading process. Each take-home exam consists of four exercises with multiple subquestions. For the first exam, we gathered 134 student submissions and two AI-generated responses, while for the second, we collected 122 student submissions along with two AI-generated responses.

Each exercise was graded independently, and we collected the data per exercise. Grading was conducted by a teaching team member unaware of the submission source (student or AI), ensuring a blind evaluation process. This approach ensures an unbiased comparison between AI-generated and student solutions, making the study reflective of a realistic educational setting. Our grading team consists of PhD students and top-performing bachelor’s and master’s students (top 10%) from previous years of the course. To ensure transparency, consistency, and to minimize subjectivity in grading, we implemented a rigorous multi-stage process:

³<https://inf.ethz.ch>

⁴The course also includes written in-person exams, which we excluded from evaluation.

Table 1: Prompt patterns.

Types	Prompts
Problem-Solving Request	<p>“Please try to solve the following problem. Note that there are three questions in total. Generate your answer in LaTeX codes. Your solution should be logically consistent and contain necessary steps and explanations toward the final answers. ⟨LaTeX SOURCE CODE OF THE PROBLEM STATEMENT⟩”</p> <p>“Now consider the following problem. There are 4 subquestions in total. Please take the hints into consideration and do not forget the instructions about generating your solution in LaTeX codes.”</p>
Reference to Lecture Notes	<p>“Please summarize the following Theorem 4.6 to me.” ⟨LaTeX SOURCE CODE OF THE THEOREM STATEMENT⟩</p>
Formatting Request	<p>“Please directly output LaTeX codes so that I can copy.”</p> <p>“Can you clearly state your algorithm and analyze it (still in latex codes)?”</p>

graders first drafted a grading scheme, which was reviewed by course lecturers; this was followed by a first-round grading and a second-round review conducted by PhD-level graders. After grading was completed, we revealed our experiment to all graders and asked for their permission to use the results. All graders gave permission.

AI-solution generation. For each exercise, a member of the teaching team generated solutions using both GPT-4o and o1-preview. This person did not participate in any grading. Since the problem statements are well-structured and clearly worded, no prompt engineering was applied. We used only text-based prompts, without incorporating vision inputs or attachments. Additionally, as the course lecture notes were not provided, the LLMs had to rely solely on prior knowledge from their pre-training to solve the problems. However, an exception was made for exercises that explicitly required the use of certain results from the lecture notes. In such cases, we explicitly supplied the relevant content to ensure the model had access to the necessary information and asked the model to summarize the results to make sure it understood correctly. The detailed prompts used in this experiment are provided in [Table 1](#).

For each exercise and for each GPT model, we repeated the query process

three times and selected the best response for submission. We observed that AI-generated solutions exhibit distinct patterns and formatting, making them easily distinguishable from student submissions. To address this, we made *minimal edits* to remove obvious AI-specific artifacts while preserving the content and reasoning of the solutions. These adjustments helped prevent graders from unintentionally identifying AI-generated responses based on formatting alone. More specifically, our edits include:

1. Removing repeated problem statements, as GPT often restates the question before providing an answer;
2. Eliminating excessive headings and subheadings, since GPT frequently structures responses with multiple hierarchical levels;
3. Omitting unnecessary thought processes, as GPT sometimes explicitly outlines its reasoning before presenting the actual solution;
4. Reformatted bullet points into standard paragraphs to make the writing style more natural and human-like.⁵

In our experiment, we found that only minimal edits to AI-generated responses were needed to support effective blind grading. In proof-based mathematical problems, student submissions naturally vary in structure and formatting due to differences in reasoning style and experience. This natural diversity in student submissions reduces the likelihood that AI-generated solutions stand out, even if with slight presentation deviation from typical answers.

4 Results

In this section, we start with a coarse-grained analysis of score distributions, followed by a fine-grained examination of error patterns based on graders' comments. Additionally, we conducted supplementary experiments using exercises from previous years to further validate our findings.

4.1 Score Comparison

Section 4.1 illustrates the score information for eight exercises across two exams, highlighting variability in performance among students and AI models (GPT-4o and o1-preview). The horizontal axis represents the exercises, while the vertical axis indicates the percentage of full scores achieved. Each exercise's score distribution

⁵Exceptions were made when bullet points are a natural way to present the content.

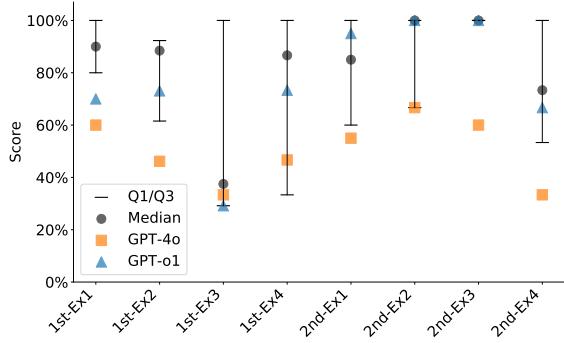


Figure 1: Score per exercise.

is visualized with three key quantile markers: the .25 quantile/Q1 (lower marker), the .5 quantile/median (middle marker), and the .75 quantile/Q3 (upper marker).

The data reveals significant differences in score distributions across exercises, reflecting differences in difficulty levels. For example, in 1st-Ex3, the range between the .25 and .75 quantile is wide, indicating a high level of difficulty. Students' scores are widely dispersed, with some struggling while others excelled. In contrast, for 2nd-Ex3, over 75% of students achieved full points, suggesting that this exercise was relatively easy for most students.

GPT-4o consistently performs below or slightly above the .25 quantile. This indicates significant challenges for GPT-4o in tackling the exercises effectively. Except for 1st-Ex3 where both o1-preview and GPT-4o performed badly, o1-preview constantly outperforms GPT-4o, even by a substantial margin. Notably, in the second exam, o1-preview demonstrates particularly strong performance, exceeding the median and achieving scores close to full marks for 2nd-Ex1, 2nd-Ex2, and 2nd-Ex3.

Exam passing score. Aggregating the scores across all exercises in each exam provides insights into the AI's ability to meet the passing criteria. [Table 2](#) summarizes the results.

For the first exam, where the passing score is set at 50% of the total score across the four exercises, GPT-4o obtained 46%, falling short of the passing threshold and outperforming only 17.65% of students. In contrast, o1-preview achieved 62%, exceeding the passing threshold and outperforming 36.03% of students.

For the second exam, given a relatively easier problem set, the passing score is higher, set at 60% of the total score across the four exercises. GPT-4o obtained 56%, falling short of the passing threshold and outperforming only 6.45% of students. In

Table 2: Exam Passing Information

Exam	Passing	GPT-4o		o1-preview	
		score	quantile	score	quantile
First	50	46	17.65%	62	36.03%
Second	60	56	6.45%	92	58.07%

contrast, o1-preview achieved an impressive 92%, exceeding the passing threshold by a large margin and outperforming 58.07% of students.

This analysis highlights the clear superiority of o1-preview over GPT-4o in both of the two exams and demonstrates the variability in their relative performances depending on the difficulty of the exercises and the context of the exam.

4.2 Error Types

Through the above score comparison, we conclude that o1-preview outperforms GPT-4o. In particular, o1-preview has the capability to pass the university-level course with a relatively large margin, while GPT-4o slightly falls below the passing score. In this part, we conduct a fine-grained analysis of each exercise. Based on the graders' feedback and comments, we draw some interesting error patterns and findings, and deep dive into the problem-solving abilities of AI tools.

We categorize the errors identified by graders into three major types:

1. **Unjustified Claims:** Arguments that appear plausible but lack sufficient rigor or proper justification.
2. **Misleading Claims:** Incorrect or invalid arguments presented as valid reasoning.
3. **Mathematical Errors:** These include basic arithmetic or algebraic mistakes, incorrect applications of theorems, failure to address edge cases, misuse of technical terms, and related issues.

The error types identified in solutions generated by GPT-4o and o1-preview are summarized in [Table 3](#). In [Section 6](#) and in the appendix of the full version of this paper [7], the error types and graders' feedback for each AI-generated solution are highlighted in red.

When analyzing the error patterns of GPT-4o and o1-preview, we observe clear differences in their strengths and weaknesses across multiple categories of errors.

Table 3: Error types per exercise.

Exercise	Unjustified Claims		Misleading Claims		Mathematical Errors	
	GPT-4o	o1-preview	GPT-4o	o1-preview	GPT-4o	o1-preview
1st-Ex1	×				×	×
1st-Ex2	×		×		×	×
1st-Ex3	×		×	×		
1st-Ex4	×	×	×		×	×
2nd-Ex1	×	×	×		×	
2nd-Ex2	×		×			
2nd-Ex3			×			
2nd-Ex4	×	×			×	

Unjustified Claims. One of the most prominent issues with GPT-4o is its frequent tendency to make unjustified claims, which occurs in seven out of eight exercises. This suggests that GPT-4o often generates plausible-sounding but ultimately unsupported reasoning. In contrast, o1-preview exhibits this error significantly less frequently, appearing in only three exercises.

A key reason for this discrepancy is that o1-preview produces longer, more structured, and more detailed responses compared to GPT-4o. While verbosity does not necessarily equate to correctness, it appears to help o1-preview provide better justification and avoid making claims without supporting arguments.

Misleading Claims. Another major issue with GPT-4o is its tendency to make misleading claims, which appear significantly more frequently compared to o1-preview. These errors typically involve incorrect arguments presented in a confident manner, making them particularly problematic for grading, as they may appear superficially correct but lack validity upon closer inspection.

However, o1-preview is not entirely free from this issue either. In 1st-Ex3 (Section 6.2), both GPT-4o and o1-preview adopt the same incorrect approach. The significant underperformance compared to students might be AI models' lack of familiarity with the course content compared to the students. Moreover, the inability to reason effectively about geometric concepts may also explain this outcome. Solving 1st-Ex3 requires spatial intuition and geometric reasoning—areas where current LLMs tend to struggle [15, 19].

Mathematical Errors. Both GPT-4o and o1-preview are susceptible to mathematical errors, but GPT-4o commits these errors more frequently and with greater severity. This suggests that GPT-4o may be less precise in its numerical and algebraic reasoning, leading to more fundamental mistakes in problem-solving.

In contrast, o1-preview’s mathematical errors tend to be more localized and less impactful on the overall correctness of its solutions. For instance, in subquestion (b) of 1st-Ex2 ([Section 6.1](#)), o1-preview misinterprets notation, leading to minor computational errors. However, despite these small mistakes, the overall reasoning and logical approach remain intact, meaning the solution still demonstrates a good understanding of the underlying concepts. This is in stark contrast to GPT-4o, which sometimes fails in a more fundamental way by misapplying mathematical principles altogether.

Other Findings. Course instructors often use the strategy of camouflaging existing problems to prevent students from directly finding solutions online when designing new questions. For example, in 2nd-Ex2 ([Section 6.3](#)), students were tasked with analyzing a linear programming technique used in designing an approximation algorithm for the Boolean satisfiability problem (CNF-SAT). To obscure its connection to well-documented materials, the instructor reframed the problem in the context of circuit analysis, substituting key terms with alternative terminology. Specifically, “CNF formula” became “NOT-OR circuit”, “literal” was replaced with “input wire”, “clause” became “intermediate wire”, and “number of satisfied clauses” was termed “output current”.

Despite these deliberate modifications, o1-preview successfully identified the underlying mathematical structure and provided a strong solution, whereas GPT-4o struggled. One possible interpretation of this is that o1-preview is more adept at leveraging pre-existing knowledge from publicly available sources, potentially explaining its superior performance on problems with well-documented solutions.

While we do not have conclusive empirical evidence to support this hypothesis, our results represent a small but meaningful step toward understanding the true nature of LLMs’ problem-solving abilities: whether they stem from genuine reasoning or from leveraging correlations and latent patterns acquired during training. This remains a fundamental and open research question in the study of language models.

5 Discussion

In this section, we share grader reflections after revealing the GPT experiment to them and discuss potential implications for university educators. We also discuss several limitations of our study.

5.1 Grader Reflections

After completing the experiment, we informed the graders about the study, and several shared their experiences evaluating AI-generated solutions. Surprisingly, except for the grader of 1st-Ex4, who initially expressed suspicion while grading, all other graders stated they could not distinguish AI-generated responses from student submissions, even after being told which ones were AI-generated. This underscores the effectiveness of our minimal editing approach in making AI-written solutions to resemble student writing styles.

More broadly, graders consistently identified “misleading claims” as a major challenge in evaluating AI-generated free-response answers, particularly for complex reasoning tasks. Unlike clearly incorrect responses, misleading claims often appear well-structured and confident, making them deceptively plausible. This forced graders to actively disprove incorrect but superficially logical arguments, which was significantly more time-consuming and cognitively demanding than simply verifying a correct solution.

Implications for university educators. Given the strong performance of LLMs on university-level assignments, we recommend gradually reducing the weight of take-home exams and supplementing them with in-class assessments. In the future, take-home assignments may be more effective when used as *formative* rather than *summative* assessments.

Additionally, it is important to adopt AI-aware evaluation strategies that emphasize transparent and well-justified reasoning. For instance, our analysis of AI-generated responses reveals that LLMs frequently make unjustified or misleading claims. Grading rubrics can be adapted to penalize these types of errors more heavily, helping to distinguish genuine human reasoning from AI-generated artifacts.

The growing capabilities of LLMs also pose new challenges for question design. Simple “camouflaging” of existing problems is no longer sufficient to prevent high-performing LLMs from generating plausible answers. Instructors may need to craft questions that target areas where LLMs still struggle, such as multi-step logical analysis.

5.2 Limitations

Dataset size. Our dataset is relatively small, consisting of 8 exercises with 20 subquestions. Given the limited sample size, the variance of AI-generated solutions may be high, which could introduce some uncertainty in the observed patterns.

However, we emphasize that our dataset is of exceptionally high quality, requiring substantial effort in both exercise design and solution evaluation. The careful

construction of this dataset makes it a valuable resource for evaluating LLMs in an academic setting, and we believe sharing it for further experimentation will contribute meaningfully to the field.

Non-penalized errors. The grading schemes used in this study may not capture all errors made by GPT models. In particular, for “unjustified claims”, graders did not always deduct points when such reasoning issues arose, as they were not explicitly penalized under the grading scheme. Instead, graders provided written comments highlighting these issues. As a result, there may be a slight discrepancy between the numerical scores and the fine-grained error analysis.

Despite this, the grading scheme was applied consistently across all student and AI-generated submissions, ensuring that the comparative evaluation between students, GPT-4o, and o1-preview remains reliable. The overall findings still provide meaningful insights into AI performance in solving complex academic problems.

Potential underestimation of LLMs’ capabilities. When generating AI solutions, we deliberately used naive prompts—simple, zero-shot inputs without advanced prompt engineering—to avoid human bias. While this approach may not fully activate the reasoning capabilities of LLMs, it aligns with the intended purpose of our benchmark: not to probe the upper limits of LLM performance, but to establish a realistic lower bound on their effectiveness in educational contexts under minimal prompting.

Additionally, since the evaluated tasks were take-home assignments, students may have had access to AI tools and potentially interacted with them using more advanced strategies, such as chain-of-thought (CoT) prompting, which can enhance performance. While this may have inflated student scores, it does not undermine the validity of our assessment of LLM performance. On the contrary, it strengthens our conclusions about o1-preview’s capabilities: its ability to achieve comparable or even superior scores in such a setting highlights its problem-solving abilities and underscores the need for reassessing how student performance is evaluated in the presence of powerful LLMs.

6 Selected Exercises

In this part, we include several exercises from our dataset to illustrate and compare the performance of GPT-4o and o1-preview. The complete dataset is available in the appendix of the full version of the paper [7].

6.1 1st-Ex2: Number of Spanning Trees in a Complete Graph

The objective of the exercise is to find the number of distinct spanning trees in a complete graph with n vertices labeled with n distinct labels. An edge between vertices i and j is labeled $\{i, j\}$, and two spanning trees are considered different if they contain any differently labeled edges. For example, on the graph with $n = 3$ labeled vertices, we have 3 distinct spanning trees.

Let $K_n = (V, E)$ be the complete graph with n vertices and suppose that the vertices are labeled $1, 2, \dots, n$. Let $R \subseteq V : |R| = k$ be a set of k vertices that are fixed (for simplicity, you can think of $R = \{1, 2, \dots, k\}$). Denote with $T_{n,k}$ the number of (labeled) forests on $\{1, \dots, n\}$ consisting of k trees whose roots are the vertices in R . Note that $T_{n,k}$ does not depend on R but only on its size and by fixing the roots, the number of possible forests decreases, eg. $T_{3,2} = 2$ as the vertices 1 and 2 cannot be in the same tree (using $R = \{1, 2\}$). For coherence, define also $T_{0,0} = 1$ and $T_{n,0} = 0$ for $n > 0$.

(b) Show that for all $1 \leq k \leq n$,

$$T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}.$$

Hint: It might be helpful to use different sets R in recursive cases.

GPT-4o	o1-preview
We fix one vertex (say, vertex 1), and consider different ways in which this vertex can be part of a forest: Vertex 1 could either belong to one of the k roots or a subtree of any other root. If vertex 1 belongs to a tree rooted in R (with $ R = k$), we recursively consider the remaining $n - 1$ vertices and form $k - 1 + i$ disjoint trees. [Unjustified Claims: Why $k - 1 + i$? Not explained] The number of ways to choose i vertices from the remaining $n - k$ vertices to add to the forest is given by $\binom{n-k}{i}$.	<p>We aim to establish a recursive relationship for $T_{n,k}$ by considering the role of vertex n in the forests. Let $S = V \setminus R$ be the set of non-root vertices, with $S = n - k$. We consider how vertex n can be connected in the forest: Since R is fixed and n may or may not be in R, but for consistency, we can assume R does not contain n (if $n \in R$, the argument is similar). [Mathematical Errors: No, but you can assume without loss of generality.] Vertex n is not a root (since $n \notin R$), so it must be connected to some other vertex in the forest. We consider all possible ways to connect vertex n:</p> <ul style="list-style-type: none"> • Connecting to a Root in R: Vertex n connects directly to one of the k roots in R. This does not change the number of trees in the forest. However, since R is fixed, we cannot choose which roots are in R, so we proceed differently.

Continued on the next page...

GPT-4o	o1-preview
<p>Thus, the recurrence relation is:</p> $T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}$ <p>[Unjustified Claims: Why $n - k$? Not explained.]</p>	<ul style="list-style-type: none"> Connecting to Non-root Vertices in S: Vertex n connects to one or more [Mathematical Errors: Should be “zero or more”.] vertices in S. By connecting vertex n to some vertices in S, we might merge their subtrees, possibly reducing the number of trees in the forest. <p>We proceed by considering the number of ways vertex n can be connected to vertices in S:</p> <ul style="list-style-type: none"> Choose i vertices from S to connect to n: There are $\binom{n-k}{i}$ [Mathematical Errors: Should be $\binom{n-k-1}{i}$.] ways to choose i vertices from S. Resulting Number of Trees: By connecting n to these i vertices, we potentially reduce the number of trees by i. However, since n is not a root, the total number of roots remains k. So after removing vertex n, the forest consists of $n-1$ vertices and $k-1+i$ trees (since i subtrees may have been merged) [Mathematical Errors: i in our formula $k - 1 + i$ is $\deg(n)$ but $i + 1$ is $\deg(n)$ in your solution.]. <p>Therefore, for each i from 0 to $n - k$, the number of forests is:</p> $\binom{n-k}{i} T_{n-1,k-1+i}.$ <p>Summing over all possible i, we obtain:</p> $T_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} T_{n-1,k-1+i}.$

6.2 1st-Ex3: Point Location

Given a collection S of n points in the plane and a constant $d > 0$, consider the problem of, given a query point q , finding all the points in S that are at most at distance d from q . Devise a data structure for this problem. In order to get the maximum score, the preprocessing time should be polynomial in the number of points n and if k is the number of points to report, the query should take $O(\log(n)+k)$ operations in expectation.

GPT-4o	o1-preview
<p>[Misleading Claims: Wrong approach.]</p> <p>We propose using a k-d tree (k-dimensional tree) to solve this problem. A k-d tree is a binary tree that recursively partitions the points based on their coordinates. The k-d tree is built recursively as follows:</p> <ul style="list-style-type: none"> • Start with the set of points S and partition it based on one of the coordinates (either x or y). • At the root level, partition the points by the x-coordinate: choose the median of the x-coordinates as the splitting value, and divide the points into two sets: those with x-coordinates smaller than or equal to the median, and those with larger x-coordinates. • Recursively repeat this process for the left and right subtrees. Alternate between splitting by x-coordinate and y-coordinate at successive levels of the tree. <p>Time complexity for building: Building the k-d tree takes $O(n \log n)$, because at each level of the recursion, we sort the points and split them, and there are $\log n$ levels in a balanced tree. To query for all points within a distance d from a given point q, we perform a range search on the k-d tree:</p> <ul style="list-style-type: none"> • Start at the root of the tree and compare the query point q with the splitting coordinate at the current level (either x or y). • If the distance from q to the splitting plane is less than d, recursively search both the left and right subtrees. • If the distance from q to the splitting plane is greater than d, only search the subtree on the side where q lies. • At each leaf node, compute the Euclidean distance between the point and q. If this distance is less than or equal to d, report the point. <p>Time complexity for querying: The query time is $O(\log n + k)$, where $\log n$ comes from traversing the tree and k is the number of points reported.</p> <p>[Unjustified Claims: Not obvious argument.]</p>	<p>[Misleading Claims: Wrong approach.]</p> <p>We propose using a balanced k-d tree (k-dimensional tree), which is a binary search tree optimized for organizing points in a multidimensional space—in this case, two dimensions. We build the k-d tree by recursively partitioning the set S as follows: At each level of the tree, we choose a splitting dimension. For a two-dimensional plane, we alternate between the x-coordinate and the y-coordinate. We select the median point along the chosen dimension to split the set. This ensures the tree remains balanced, resulting in a depth of $O(\log n)$. The points are divided into two subsets: those with coordinates less than or equal to the median and those greater than the median, along the splitting dimension. This process continues recursively until each leaf node contains a single point or a small number of points. The preprocessing time to build the k-d tree is $O(n \log n)$, as sorting is required at each level to find medians.</p> <p>Query Algorithm: To find all points within distance d from a query point q, we perform a range search on the k-d tree: We start at the root of the tree and compare q's coordinate along the splitting dimension with the node's splitting value. If the absolute difference between q's coordinate and the splitting value is less than or equal to d, both subtrees might contain points within distance d of q. In this case, we recursively search both subtrees. If the difference is greater than d, only one subtree needs to be searched—the one on the same side as q. At each node, if the point stored there is within distance d from q, we add it to the list of points to report. The search continues recursively until all relevant subtrees have been explored. The traversal time through the tree is $O(\log n)$ for a balanced k-d tree. [Misleading Claims: Not true if we go through more branches.] The time to report the k points within distance d of q is $O(k)$, since we check each candidate point individually. Therefore, the total expected query time is $O(\log n + k)$.</p>

6.3 2nd-Ex2: NOT-OR Circuits Using Linear Programming

A digital circuit consists of a network of wires connected through logic gates. Each wire carries a signal, which can be either 0 (representing *false*) or 1 (representing *true*). In this exercise, we consider a class of circuits called *NOT-OR circuits*, which consist of three sequential layers.

- **Input Layer:** The circuit has n input wires, where each wire i (for $i = 1, 2, \dots, n$) carries either a 1 or a 0. The sequence of values on these input wires is called the *input configuration*.
- **Negation Layer:** The n input wires are passed through a layer that outputs $2n$ wires. For each input x_i , this layer produces both x_i and its negation, denoted as $\text{not}(x_i)$. These values are referred to as *intermediate wires*.
- **OR Layer:** The intermediate wires are then fed into a layer that outputs m wires, each representing a *OR-gate* value $C_j \in \{0, 1\}$ computed from a subset of the intermediate wires (for $j = 1, 2, \dots, m$). Each OR-gate value is 1 if at least one of its intermediate wires has value 1. Multiple OR-gates may receive the same intermediate wire as input.

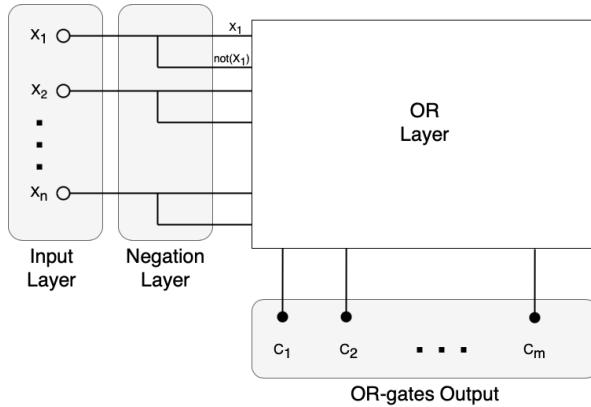


Figure 2: A scheme of a NOT-OR circuit

We consider a NOT-OR circuit where each OR-gate may take as input an arbitrary number of intermediate wires. Each output OR-gate C_j controls a switch that regulates an output current $f_j \geq 0$. The *total output* of the circuit is defined as $\sum_{j=1}^m C_j f_j$. Given a circuit and outputs f_1, \dots, f_m , our goal is to find an input configuration that maximizes this total output.

(b) Design a polynomial-time deterministic algorithm that returns an input configuration achieving at least half of the maximum possible output current.

Hint: Consider an input configuration and its negation.

GPT-4o	o1-preview
<p>Algorithm: We aim to design a polynomial-time deterministic algorithm that achieves at least half of the maximum possible output current.</p> <p>Algorithm: Given a NOT-OR circuit with n input wires, proceed as follows: 1. Compute the output current for the given input configuration x. 2. Compute the output current for the negation of the input configuration $\text{not}(x)$. 3. Select the configuration (either x or $\text{not}(x)$) that results in the higher output current. [Unjustified Claims: Needs more argumentation, you are basically already assuming the thing you want to show here.]</p> <p>Justification: Each OR-gate C_j is connected to the values of the intermediate wires x_i and $\text{not}(x_i)$. Therefore, either x or $\text{not}(x)$ must contribute at least half of the maximum output current. Since the algorithm evaluates both configurations, it achieves at least half of the maximum possible output current.</p> <p>Conclusion: This deterministic algorithm runs in polynomial time and guarantees at least half of the maximum output current. [Misleading Claims: Wrong claim.]</p>	<p>Algorithm: 1. Compute the total output when all input wires are assigned 0. Let this output be T_0.</p> <p>2. Compute the total output when all input wires are assigned 1. Let this output be T_1.</p> <p>3. Choose the input configuration (all zeros or all ones) that yields the higher total output.</p> <p>Proof of correctness: Consider any input configuration S and its negation \bar{S} (where each input is flipped). For each OR-gate C_j, at least one of C_j and \bar{C}_j outputs 1 because the inputs to C_j in S are the negation of its inputs in \bar{S}. Therefore:</p> $C_j + \bar{C}_j \geq 1.$ <p>Let T_{\max} be the maximum possible total output. Then:</p> $\begin{aligned} T(S) + T(\bar{S}) &= \sum_{j=1}^m (C_j + \bar{C}_j)f_j \\ &\geq \sum_{j=1}^m f_j \geq T_{\max}. \end{aligned}$ <p>Therefore, at least one of $T(S)$ or $T(\bar{S})$ is at least $\frac{1}{2}T_{\max}$.</p>

References

- [1] Harun Bayer, Fazilet Gü Ince Araci, and Gülsah Gürkan. Chatgpt-4o, chatgpt-4 and google gemini are compared with students: A study in higher education. *International Journal of Technology in Education and Science*, 8(4):627–644, 2024.
- [2] Dana Brin, Vera Sorin, Akhil Vaid, Ali Soroush, Benjamin S Glicksberg, Alexander W Charney, Girish Nadkarni, and Eyal Klang. Comparing chatgpt and gpt-4 performance in usmle soft skill assessments. *Scientific Reports*, 13(1):16492, 2023.

- [3] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A Survey on Evaluation of Large Language Models. 15(3):1–45.
- [4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code.
- [5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.
- [6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems.
- [7] Ming Ding, Rasmus Kyng, Federico Solda, and Weixuan Yuan. Assessing gpt performance in a proof-based university-level course under blind grading. *arXiv preprint arXiv:2505.13664*, 2025.
- [8] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.
- [9] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.
- [10] GitHub. GitHub Copilot: Your AI Pair Programmer, 2024.
- [11] Google DeepMind. Gemini AI, 2024.
- [12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

- [13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset.
- [14] Steffen Herbold, Annette Hautli-Janisz, Ute Heuer, Zlata Kikteva, and Alexander Trautsch. A large-scale comparison of human-written versus chatgpt-generated essays. *Scientific reports*, 13(1):18617, 2023.
- [15] Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricu. Geomverse: A systematic evaluation of large models for geometric reasoning. *arXiv preprint arXiv:2312.12241*, 2023.
- [16] Markus Kipp. From gpt-3.5 to gpt-4. o: A leap in ai's medical exam performance. *Information*, 15(9):543, 2024.
- [17] Ehsan Latif, Yifan Zhou, Shuchen Guo, Lehong Shi, Yizhu Gao, Matthew Nyaaba, Arne Bewerdorff, Xiantong Yang, and Xiaoming Zhai. Can OpenAI o1 outperform humans in higher-order cognitive thinking?
- [18] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic Search-Enhanced Large Reasoning Models.
- [19] Spyridon Mouselinos, Henryk Michalewski, and Mateusz Malinowski. Beyond lines and circles: Unveiling the geometric reasoning gap in large language models. *arXiv preprint arXiv:2402.03877*, 2024.
- [20] Sasha Nikolic, Carolyn Sandison, Rezwanul Haque, Scott Daniel, Sarah Grundy, Marina Belkina, Sarah Lyden, Ghulam M Hassan, and Peter Neal. Chatgpt, copilot, gemini, scispace and wolfram versus higher education assessments: an updated multi-institutional study of the academic integrity impacts of generative artificial intelligence (genai) on assessment, teaching and learning in engineering. *Australasian journal of engineering education*, 29(2):126–153, 2024.
- [21] OpenAI. GPT-4 Technical Report.
- [22] OpenAI. ChatGPT: Optimizing Language Models for Dialogue, 2024.
- [23] Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating llms on 2025 usa math olympiad. *arXiv preprint arXiv:2503.21934*, 2025.
- [24] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- [25] Mike Richards, Kevin Waugh, Mark Slaymaker, Marian Petre, John Woodthorpe, and Daniel Gooch. Bob or Bot: Exploring ChatGPT's Answers to University Computer Science Assessment. 24(1):1–32.
- [26] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Wino-Grande: An adversarial winograd schema challenge at scale. 64(9):99–106.

- [27] SciSpace. SciSpace: AI-Powered Research Platform, 2024.
- [28] Daniel Stribling, Yuxing Xia, Maha K Amer, Kiley S Graim, Connie J Mulligan, and Rolf Renne. The model student: Gpt-4 performance on graduate biomedical science exams. *Scientific Reports*, 14(1):5670, 2024.
- [29] Teo Susnjak and Timothy R McIntosh. Chatgpt: The end of online exam integrity? *Education Sciences*, 14(6):656, 2024.
- [30] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- [31] Esteban Vázquez-Cano, Jose M Ramirez-Hurtado, Jose M Saez-Lopez, and Eloy Lopez-Meneses. Chatgpt: The brightest student in the class. *Thinking Skills and Creativity*, 49:101380, 2023.
- [32] Jonathan Vidal. Evaluation of the performance of state-of-the-art large language models (llms) in solving math word problems. *Available at SSRN 4902960*, 2024.
- [33] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 36.
- [34] Will Yeadon, Alex Peach, and Craig Testrow. A comparison of human, gpt-3.5, and gpt-4 performance in a university-level coding course. *Scientific Reports*, 14(1):23285, 2024.
- [35] Roozbeh Yousefzadeh, Xuenan Cao, and Azim Ospanov. A lean dataset for international math olympiad: Small steps towards writing math proofs for hard problems. *arXiv preprint arXiv:2411.18872*, 2024.
- [36] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

The Bulletin of the EATCS

The Bulletin of the EATCS

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

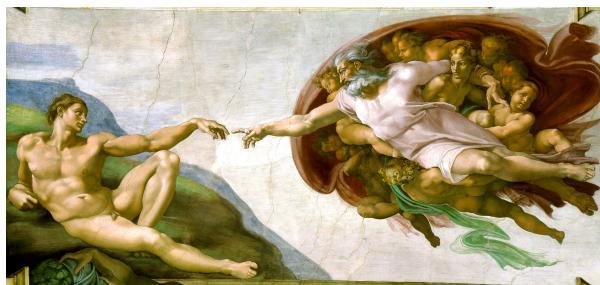
YURI GUREVICH

Computer Science & Engineering
University of Michigan, Ann Arbor, Michigan, USA
gurevich@umich.edu

INTERACTIVE CLASSICAL ALGORITHMS: PREVIEW

Yuri Gurevich

June 2025



The power of interaction

Abstract

This dialog paper offers a preview and provides a foretaste of an upcoming work on the axiomatization of interactive classical algorithms.

The modern notion of algorithm was elucidated in the 1930s–1950s. It was axiomatized a quarter of a century ago as the notion of “sequential algorithm”; we call it “classical algorithm” here. The axiomatization was used to show that for every classical algorithm there is a behaviorally equivalent abstract state machine. It was also used to prove the Church-Turing thesis as it has been understood by the logicians.

Starting from the 1960s, the notion of algorithm has expanded — probabilistic algorithms, quantum algorithms, etc. — prompting introduction of a much more ambitious version of the Church-Turing thesis commonly known as the “physical thesis.” We emphasize the difference between the two versions of the Church-Turing thesis and illustrate how nondeterministic and probabilistic algorithms can be viewed as classical algorithms with appropriate oracles. The same view applies to quantum circuit algorithms and many other classes of algorithms.

⁰Partially supported by the US Army Research Office under W911NF-20-1-0297.

1 Is coin flipping algorithmic?

The Bulletin of the EATCS

Q: Consider flipping a coin. Is it algorithmic?

A: Why do you ask?

Q: People speak about randomized algorithms, involving probability distributions. These distributions stem from physical processes, like flipping a coin. So a question arises: *who* flips the coin? I don't see how an algorithm could achieve that. An external agent has to perform the flip.

A: You could also ask whether a quantum measurement is algorithmic.

Q: Isn't that essentially the same question? A quantum measurement also yields a probability distribution.

A: Well, if you doubt that an algorithm can flip a coin, you might be even more skeptical about an algorithm performing a quantum measurement. That process involves Mother Nature, after all.

Q: But coin flipping involves Mother Nature as well, doesn't it?

A: You are right. The difference is that quantum measurement involves an aspect of nature that we are not accustomed to and don't fully understand, whereas coin flips have been well understood for a long time.

In any case, I stick to the traditional view that algorithms are inherently deterministic, making "nondeterministic algorithm" a contradiction in terms. Yogi Berra, an American philosopher and baseball player, once illustrated this: "When you come to a fork in the road, take it."

Q: How do you reconcile this view with the widespread use of the term "nondeterministic algorithm"?

A: This could be just a figure of speech. Nondeterministic algorithms can be viewed as deterministic algorithms that interact with their environment where someone makes the necessary choices, possibly by flipping a coin. Some authors say that a random sequence of 0's and 1's is part of the input, so that the necessary choices are made ahead of time.

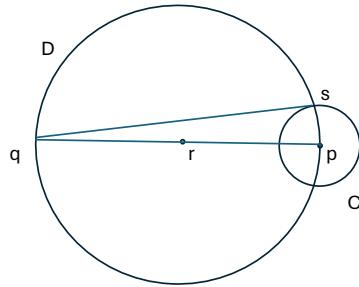
Alternatively — and quite legitimately — one can broaden the notion of algorithm, just as the notion of numbers was broadened. From positive integers all the way to real numbers, then to complex numbers and beyond.

Q: Let's consider a simple example of a nondeterministic algorithm.

BEATCS no 146
A: Here's a classic ruler-and-compass algorithm with minimal nondeterminism. The setting is a fixed Euclidean plane. Given a circle C , its center p , and a point q outside of C , the algorithm constructs a tangent from q to C .

1. draw the midpoint r between p and q ;
 2. draw the circle D centered at r and passing through q ;
 3. choose a point s where the circles C, D intersect;
 4. draw the line through q and s .
- (1)

The resulting line through q and s is the desired tangent to C .



Instruction 3 involves a nondeterministic choice. Either intersection point will work, but the choice must be made. The algorithm doesn't specify which one to choose; presumably, that's left to the executor. Aside from this step, the construction is fully deterministic.

2 Two vastly different theses

Q: Before we plunge into interactive algorithms, let me ask you, again, about the Dershowitz-Gurevich derivation of the Church-Turing thesis in [7]. Peter Shor is critical of it, but I'm interested in exploring the concerns he raises.

“The Dershowitz-Gurevich paper says nothing about probabilistic or quantum computation. It does write down a set of axioms about computation, and prove the Church-Turing thesis assuming those axioms. However, we’re left with justifying these axioms. Neither probabilistic nor quantum computation is covered by these axioms (they admit this for probabilistic computation, and do not mention quantum computation at all), so it’s quite clear to me these axioms are actually false in the real world, even though the Church-Turing thesis is probably true” [21].

But let me start with this: Do you think he read the paper?

The Bulletin of the EATCS
A: I'd guess that he just skimmed the abstract and then searched the text for "probabilistic" and "quantum."

Q: Is this arrogance or misperception?

A: Both, I think. The misperception would disappear if he read just a bit beyond the abstract.

There are two vastly different interpretations of the Church-Turing thesis in play. The thesis may be formulated thus:

Church-Turing thesis. *Every effectively calculable string-to-string function is Turing computable.*

A question arises: what does "effectively calculable" mean?

One interpretation is traditional in logic. In the 1930s – 1950s, the meaning of "effectively calculable" was not in dispute. Logicians, including Church and Turing, — Turing wrote his dissertation in logic, under Church — had robust intuition about it. This intuition is elucidated in many books, for example in the influential books [15, §62], [19, §1.1], and [20, §9]. In this interpretation, effective calculation was deterministic. For example Turing writes: "The behaviour of the computer at any moment is determined by the symbols which he is observing, and his 'state of mind' at that moment" [22, §9]. Let me call this original interpretation of the thesis "classical" (though we should be careful with this term because, in quantum computing, "classical" means "not quantum").

The other interpretation is also natural in a sense, especially if you don't know the history of the subject. "Effectively calculable" can be interpreted as physically computable, which allows probabilistic and quantum computations as well as highly parallel, distributed, etc. This broader interpretation is commonly known as the "physical thesis."

Q: I see. You derive the classical thesis, and you think that Shor has in mind the physical thesis.

A: Yes, that is what I think. §1 of our paper is about effectivity. If Shor read just a little beyond the abstract, he would know that we prove the classical thesis.

Q: Why is it "quite clear" to Shor that your axioms "are actually false in the real world?"

A: I can only speculate about that. The axioms fail to cover probabilistic and quantum algorithms, so they are wrong and the proof does not establish the thesis. Nevertheless, the thesis "is probably true." Of course I think that it cannot be true [12].

Q: The speculation seems reasonable. But if the axioms imply the physical thesis then the thesis must have been formulated in mathematical terms, which is of independent interest. That should have occurred to Shor unless indeed he spent infinitesimal time on your paper. Anyway, is there a convincing formulation of the physical thesis?

A: I don't think so. Early on, Robin Gandy attempted such a formulation [6] but the attempt was not successful. In particular, it didn't cover distributed computing.

Q: Are your axioms true in the real world?

A: A better question is what algorithms satisfy the axioms?

Q: Do you admit that your axioms don't cover probabilistic computations?

A: "Admit" is the wrong word. Probabilistic algorithms are out of scope of our paper because they aren't classical. The relevant paragraph says: "Methods satisfying the Sequential Postulates include ... On the other hand, the postulates exclude ... They are also meant to exclude nondeterministic methods ..., probabilistic methods (like Rabin's algorithm for testing primality), ..." But the same paragraph includes this footnote:

"Large classes of such non-classical algorithms are covered by the generalizations of the ASM Theorem in [1, 2, 3, 8]."

Q: I know from our earlier conversations that "ASM" stands for "abstract state machine." But what do you mean by the ASM theorem?

A: Let's take a quick dive into the history of the ASM project. It started in mid 1980s as a computer theory project — I wanted to understand what algorithms are. That led me to the notion of abstract state machines, originally called evolving algebras, and to the

ASM thesis. For every algorithm there is a behaviorally equivalent ASM.

which I see as a natural extension of Turing's approach to the Church-Turing thesis. The project quickly became applied and practical. An ASM community emerged, and ASMs were successfully used for high-level executable specifications and related tasks. Article [14] is one example.

The ASM thesis was first formulated for classical algorithms [14] and then extended to algorithms in general [10]. Article [11] turned the ASM thesis, restricted to classical algorithms, called the Sequential ASM thesis in [11], into a theorem. That's the ASM theorem that you asked about.

Q: Why “sequential” rather than “classical” thesis? *The Bulletin of the EATCS*

A: At the time, it seemed to me that the thesis can’t be called classical, because it was new. Hence “sequential” as an imperfect substitute, emphasizing the difference from parallel, distributed, etc.

Q: But the ordinary meaning of sequential algorithms does not rule out probabilistic algorithms for example.

A: True. It would probably have been better to call the thesis classical — even though it was new — because it is about classical algorithms.

Q: Are readers “left with justifying” your axioms?

A: No. There are four axioms (or postulates) in [7]. The first three aim to capture all classical algorithms, whether they compute a function or not.

1. Sequential Time Postulate is self-evident, almost trivial. It says that an algorithm could be viewed as (finite or infinite) automaton with states, initial states, and a transition function.
2. Abstract State is natural, at least to logicians. The main part of it is that states can be viewed as structures in the sense of mathematical logic.
3. Bounded Exploration Postulate says that, during a step, only a bounded part of the state is explored, namely the part given by a fixed set of expressions.¹

Besides, the work by the ASM community provided plenty of justification. There is little doubt that classical algorithms satisfy the three axioms. The surprising part is that the axioms are sufficient to capture the notion of classical algorithms mathematically.

Q: What is the fourth axiom for?

A: A string-to-string function, computed by a classical algorithm, is not necessarily Turing computable because the initial state may have too much information. The fourth axiom guarantees that the initial states are bare.

¹The first two postulates occurred to me right away but it took me years to arrive at the third postulate. I might have arrived at it earlier if I knew at the time about Kolmogorov’s insight “An algorithmic process breaks down into separate steps of a priori bounded complexity” [16] in the minutes of Moscow Mathematical Society in *Uspekhi Matematicheskikh Nauk*, which is translated as Russian Mathematical Surveys, but minutes aren’t translated.

3^{BEATCS no 146} Spec Explorer

Q: Do you describe ASM applications in the Dershowitz-Gurevich paper?

A: We probably should have, but we didn't — mainly because we submitted the paper to the Bulletin of Symbolic Logic rather than a computer science journal.

Q: What would be a good example of ASM applications?

A: Let me tell you about Spec Explorer. The story illustrates how theoretical work on ASMs had profound practical impact.

In 1998, Microsoft Research (MSR) invited me to create a group and apply the ASM method. My first hire was Wolfram Schulte, one of the most talented people in the ASM community in Germany. Unlike me, he had industrial experience. We built a wonderful Foundations of Software Engineering group. In a few years the group built an ASM-based tool, Spec Explorer that allowed us to write high-level executable specifications and test them against programs. Upper management, including and especially Bill Gates, liked the tool. But it seemed impossible to get the developers to use the tool because it required nontrivial training. Spec Explorer was an advanced but niche tool, rich in features, beloved by testers and those who appreciated formal methods.

Then the European Union came to our rescue ^②. In the early 2000s, the EU reprimanded Microsoft—and for good reason. While outside developers were confined to official Windows interfaces, Microsoft's own products could tap into undocumented internal protocols that offered privileged access to the core of the operating system. This asymmetry made meaningful competition on the Windows platform nearly impossible².

The EU demanded change. Specifically, it required Microsoft to produce *high-level, executable* specifications of the internal Windows protocols that its own products used. Microsoft lacked comprehensive, precise, high-level executable specifications. Word documents and ad hoc specs weren't sufficient.

This created a moment of crisis. And then, unexpectedly, Spec Explorer was thrust into the spotlight. It could produce high-level executable specs using state machines and C# annotations. It could generate test suites and simulate behaviors. It was suitable for model checking and detecting inconsistencies.

The Windows Division picked up Spec Explorer to model protocols. The tool was "dumbed down" a bit for broader adoption — many industrial teams want minimal knobs. It was also strengthened and productized to scale to the thousands of pages of protocol documentation eventually published.

²The European Commission's antitrust case against Microsoft culminated in a 2004 decision (Case T-201/04).

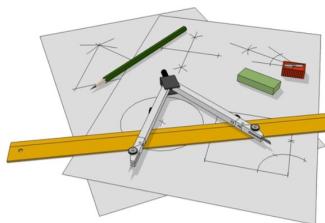
The Bulletin of the EATCS
It became an industrial-strength workhorse. Teams worked tirelessly to generate models and produce the documentation needed to satisfy the EU's demands. The specifications were released as part of the Microsoft Open Specifications Promise. Microsoft paid significant fines (over €1.6 billion across multiple years). The case marked a major shift toward transparency and openness, especially regarding protocols and APIs.

Ultimately, it was, however, a Pyrrhic victory for the EU. While Microsoft did produce thousands of pages of protocol specifications and did improve transparency, the underlying technology landscape was shifting. The relevance of those protocols was already beginning to fade. Web services, cross-platform frameworks, and cloud computing were quickly rendering the entire issue obsolete.

Still, for a brief and intense period, formal modeling and executable specifications were at the center of one of the most consequential regulatory battles in software history—and Spec Explorer played a key role in bridging the gap between legal compliance and technical execution.

4 Examples

Q: You said that classical algorithms are deterministic, but the tangent algorithm is not. It is a ruler-and-compass algorithm that comes from antiquity; it can't be more classical.



Credit: Wikipedia [23]

A: This is a matter of definition. The notion of algorithm was elucidated in logic work of the 1930s–1950s. It is these algorithms that I call classical. They are deterministic. The notion of nondeterministic algorithms was formally introduced by Stephen Cook only in 1971 [5] (and independently by Levin in 197 [17], with a slightly different terminology), though Michael Rabin and Dana Scott introduced nondeterministic finite automata in 1959 [18].

By the way, in classical geometry, the tangent algorithm is not considered nondeterministic. Both solutions are considered equally valid, and the choice is left to the user.

BEATCS no 1/6
Q: I've read Church's and Turing's thesis papers, [4] and [22], and I don't recall them using the word "algorithm."

A: You are right. The term algorithm (or algorism) existed earlier, but it referred to performing calculation using Arabic numerals. Logicians used terms like "effective method", "mechanical procedure", and "rule of calculation." It wasn't until the 1950s–60s, with the rise of computer science, that "algorithm" took on its current meaning: a finite, unambiguous, effective procedure for solving a problem.

Q: What ASM is behaviorally equivalent to the tangent algorithm?

A: Let me start from afar. Often, especially in small examples, the form of ASMs from [11] is used. It is an implicit iteration of a generic one-step rule R and is given just by R . The most common convention is that R is executed repeatedly until, if ever, you reach a fixed-point state X , so that, in state X , executing R leaves the state unchanged.

This convention is not appropriate for interactive algorithms. Executing R in a fixed-point state isn't innocent if R involves any interaction with the environment because this interaction is visible externally. Fortunately, the three axioms of [11] imply that the halting condition of a classical algorithm can be expressed by a Boolean-valued term. Accordingly, the program of a classical — in the sense of classical algorithms — ASM can be given in the form

```
do until  $H$ 
     $R$ 
```

where R is a generic step rule and H a halting condition. For example, consider the following version of Euclid's algorithm for computing the greatest common divisor $d = GCD(a, b)$ of two nonnegative integers $a \geq b$.

```
while  $b > 0$  do
    ( $a := b$ ) || ( $b := a \text{ mod } b$ );
     $d := a$                                 (2)
```

In this case, the program of a simulating ASM could be

```
do until  $d = a$ 
    if  $b > 0$  then ( $a := b$ ) || ( $b := a \text{ mod } b$ )
    else  $d := a$ 
```

where it is presumed that d is initially undefined (or has a value, like -1 , that a cannot possibly have).

Q: Typically, when you convert a while loop into a do-until loop, you just negate the condition.

The Bulletin of the EATCS
A: Since we insist that the do-until loop is the whole program, it needs to “suck in” the assignment $d := a$ and thus becomes one step longer.

Q: Implicit iteration makes good sense in this case because the step rule makes it clear when to stop

A: Indeed, if a program (i) merely computes the value of a term t that involves no oracle queries and (ii) at the final step executes an assignment $o := t$ where o is an output variable, then the obvious halting condition is $o = t$ and the do-until loop can be made implicit. In the case of Euclid’s algorithm, the program of the simulating ASM can be given by the generic step rule

```
if  $b > 0$  then  $(a := b) \parallel (b := a \bmod b)$ 
else  $d := a$ 
```

Q: Let’s return to the tangent algorithm. I’d expect that the program of an ASM that naturally simulates (1) would be similar to (1).

A: This is true in a sense, but the only ASM commands used to form a generic step rule in [11] are assignments, conditionals, and parallel compositions. Sequentiality within a step is handled by conditionals and sequentiality of different steps by iteration.

To deal with geometric objects, we use a restricted incidence relation Inc of type $\text{Point} \times \text{Circle} \rightarrow \text{Boolean}$, and we need some program variables and function symbols:

- program variables r, s of type Point , D of type Circle , and T of type Line ,
- binary function symbols C, L, M with domain type $\text{Point} \times \text{Point}$ and range types Circle , Line , and Point respectively, and
- a binary function symbol I of domain type $\text{Circle} \times \text{Circle}$ and range type Point .

The intended meaning of the binary functions is as follows. Let x, y be distinct points and A, B distinct intersecting circles. Then

- $C(x, y)$ is the circle through y with center x .
- $L(x, y)$ is the line through x and y .
- $M(x, y)$ is the midpoint between x and y .
- I is an oracle function. $I(A, B)$ is a point in the intersection of the two circles.

Making the do-until loop implicit, we can mimic the structure of (1).

```
r := M(p, q) ||
if  $r = M(p, q)$  then  $D := C(r, q) \parallel$ 
if  $D = C(r, q)$  then  $s := I(C, D) \parallel$ 
if  $\text{Inc}(s, C) \wedge \text{Inc}(s, D)$  then  $T := L(q, s)$ .  

(3)
```

BEATCS no 1/6
Q: Why isn't the guard $s = I(C, D)$ in the last line?

A: Because we cannot guarantee that the second oracle call $I(C, D)$ will give the same result as the previous one. (In [2], we have a convention that, within a step, the same queries have the same answer.)

By the way, in the case of the tangent algorithm, we can do better:

$$T := L(q, I(A, C(M(p, q), q))).$$

Q: I'd like to see an example where you start with a probabilistic algorithm, say with the Rabin Primality Test.

A: Ok, let me first recall what it is all about.

Problem: Given an integer n determine with sufficiently high probability whether n is prime.

Algorithm: Let a, i be integer variables initialized to 1, k a positive integer constant, **prime** a Boolean variable initialized to **true**, and **Random** a probabilistic oracle that, given integers $b < c$, selects a number in the segment $[b, c]$ according to the uniform probability distribution.

```
do until prime = false or i > k
  choose a random integer a such that 2 ≤ a ≤ n - 2;
  if  $a^{n-1} \neq 1$  then prime := false
  else increment i by 1
```

The probability of a false positive — n is composite but **prime** retains value **true** — decreases exponentially with k . So the desired precision can be given by the value of k .

ASM. The program of a simulating ASM could be

```
do until (prime = false) ∨ (i > k)
  a := Random(2, n - 2) || i := i + 1
  if  $a^{n-1} \bmod n \neq 1$  then prime:=false
```

Q: You said that classical/sequential algorithms are not parallel, and you keep using parallel composition in the examples.

A: In [11], parallelism is subject to the following constraints.

- It is used only within a step, and
- the number of components is a priori bounded.

Q: I thought that a single step results in at most one state changing action. Parallel actions within a single step seem to contradict that.

A: Well, even Turing machines permit these two parallel actions during a single step: changing the control state and moving the head.

5 On interactive classical algorithms

Q: You say that nondeterministic algorithms are naturally interactive. Deterministic algorithms also could be interactive.

A: Deterministic algorithms are typically interactive, though interaction is often implicit. For example, algorithm (2) doesn't have code for computing the modulo function and therefore interaction is necessary. But interaction is more problematic in the case of nondeterministic algorithms, as we saw above when we discussed (3).

Q: Have you tried to extend the formalization in [11] to allow algorithms that query possibly-nondeterministic oracles?

A: We did. The results were reported in lengthy publications [2] and [3], probably too lengthy and involved for much of the intended audience.

Q: Why so lengthy?

A: Those papers included detailed motivations, discussions of alternatives, explanations, detailed proofs. Also, they explicitly allowed viewing the operating system as part of the environment and viewing our algorithm as an agent in a distributed computation.

Anyway, currently Andreas Blass and I are finishing a much shorter version of [2] with many improvements and simplifications. The axioms are the three axioms of [11] somewhat expanded.

Q: Give me a simple example of axiom expansion.

A: The simplest example is related to the Sequential Time Postulate.

Q: I guess you just replace the transition function with a transition relation?

A: This would not be enough. There may be steps of the algorithm that transform a state X to a state X' but have different interactions with the oracles. The steps are different but a transition relation would not distinguish them.

BEATCS no 146
Q: I see. Interaction is, at least in principle, externally observable, and so the two steps exhibit different behavior.

A: Correct.

Q: Does the interactive-classical framework support quantum computing?

A: It was in fact our work on quantum circuit algorithms [13] that prompted the revision of [2]. Quantum circuits have quantum-transformation gates (unitary gates and measurement gates) but no code to perform them. Accordingly, we need oracles for those quantum transformations. Quantum circuit algorithms are naturally interactive classical/traditional algorithms.

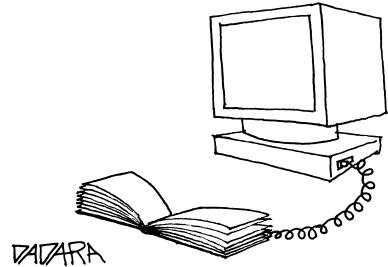
References

- [1] Andreas Blass and Yuri Gurevich, “Abstract state machines capture parallel algorithms,” *ACM Transactions on Computational Logic* 4:4 (Oct. 2003) 578–651 with “Correction and Extensions” in 9:3 (June 2008) article 19
- [2] Andreas Blass and Yuri Gurevich, “Ordinary interactive small-step algorithms”, *ACM Transactions on Computational Logic*, part I in 7:2 (April 2006), pages 363–419, parts II and III in 8:3 (July 2007) articles 15 and 16
- [3] Andreas Blass, Yuri Gurevich, Dean Rosenzweig and Benjamin Rossman “Interactive small-step algorithms,” parts I and II, *Logical Methods in Computer Science* 3:4 (2007) articles 3 and 4
- [4] Alonzo Church, “An unsolvable problem of elementary number theory”, *American Journal of Mathematics* 58 (1936), 345–363
- [5] Stephen Cook, “The complexity of theorem-proving procedures,” Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC) 1971 151—158
- [6] Robin O. Gandy, “Church’s thesis and principles for mechanisms”, In “The Kleene Symposium” (eds. J. Barwise et al.), North-Holland 1980 123–148.
- [7] Nachum Dershowitz and Yuri Gurevich, “A natural axiomatization of computability and proof of Church’s thesis,” *Bulletin of Symbolic Logic* 14:3 (2008) 299–350
- [8] Andreas Glausch and Wolfgang Reisig, “An ASM-characterization of a class of distributed algorithms,” *Lecture Notes in Computer Science* 5115, Springer, https://doi.org/10.1007/978-3-642-11447-2_4
- [9] Yuri Gurevich, “Evolving algebras: An introductory tutorial,” *Bulletin of EATCS* 43 (1991) 264–284. Also (slightly revised) in “Current Trends in Theoretical Computer Science: Essays and Tutorials” (eds. G. Rozenberg and A. Salomaa), World Scientific 1993 266–292
- [10] Yuri Gurevich, “Evolving algebra 1993: Lipari guide,” in “Specification and Validation Methods” (ed. E. Börger), Oxford University Press 1995 9–36, <https://arxiv.org/pdf/1808.06255.pdf>

- [11] Yuri Gurevich, “Sequential abstract state machines capture sequential algorithms,” *The Bulletin of the EATCS*, *ACM Transactions on Computational Logic* 1:1 (2000), 77–111
- [12] Yuri Gurevich, “Unconstrained Church-Turing thesis cannot possibly be true,” *Bulletin of EATCS* 127 (2019), <https://arxiv.org/pdf/2002.03145>
- [13] Yuri Gurevich and Andreas Blass, “Software science view on quantum circuit algorithms,” *Information and Computation* 292 (2023) article 105024, <https://arxiv.org/pdf/2209.13731>
- [14] Yuri Gurevich and Jim Huggins, “The semantics of the C programming language,” Springer Lecture Notes in Computer Science 702 (1993) 274–308.
- [15] Stephen C. Kleene, “Introduction to metamathematics,” Wolters-Noordhoff Publishing and North-Holland Publishing Company, 1971 (First published by D. Van Nostrand Company in 1952.)
- [16] Andrey N. Kolmogorov, “On the notion of algorithm,” *Uspekhi Matematicheskikh Nauk* (in Russian) vol. 8 issue 4(56) (1953) p. 175, <https://www.mathnet.ru/rus/rm/v8/i4/p173>
- [17] Leonid Levin, “Universal search problems,” *Problems of Information Transmission* (in Russian). 9:3 (1973) 115–116
- [18] Michael Rabin and Dana Scott, “Finite automata and their decision problems,” in *IBM Journal of Research and Development* 3:2 (1959) 114–125
- [19] Hartley Rogers, Jr. “Theory of recursive functions and effective computability,” McGraw-Hill 1967
- [20] Joseph R. Shoenfield, “Recursion theory,” Springer Verlag 1993
- [21] StackExchange contributors, “What would it mean to disprove Church-Turing thesis?” Theoretical Computer Science StackExchange, Asked August 17, 2010, <https://cstheory.stackexchange.com/questions/88/>
- [22] Alan M. Turing, “On computable numbers, with an application to the Entscheidungsproblem,” *Proceedings of the London Mathematical Society*, ser. 2 vol. 42 parts 3 and 4, 1936, 230–265. Corrigenda in vol. 43 (1937) 544–546.
- [23] Wikipedia, “Straightedge and compass construction,” © CC BY-SA 4.0, seen on May 15, 2025

BEATCS no 146

News and Conference Reports



DADARA

REPORT FROM EATCS JAPAN CHAPTER

Yuto Nakashima (Kyushu University)

EATCS-JP/LA Workshop on TCS and Presentation Awards

The 23rd *EATCS-JP/LA Workshop on Theoretical Computer Science* was held at Research Institute of Mathematical Sciences, Kyoto University, January 27th to 29th, 2025. (The details can also be found, although this website is written in Japanese, at

https://la-symposium.github.io/2024/la2024w_program_v3.pdf.)

Every year, we choose the best presenter and the best student presenter. This year, we celebrated the following presentation as the 23rd LA/EATCS-Japan Presentation Award:

“*Dudeney’s Dissection is Optimal*” presented by Erik D. Demaine (Massachusetts Institute of Technology), **Tonan Kamata**, and Ryuhei Uehara (Japan Advanced Institute of Science and Technology).

We celebrated the following presentation as the 14th LA/EATCS-Japan Student Presentation Award:

“*Induced Subgraph Isomorphism Reconfiguration Under Extended Reconfiguration Rules*” presented by **Tatsuhiro Suga**, Akira Suzuki, Yuma Tamura, and Xiao Zhou (Tohoku University).

The awards were recognized publicly on the last day, January 29th, 2025.

Congratulations!

This workshop is jointly organized by *LA symposium*, Japanese association of theoretical computer scientists. Its purpose is to give a place to discuss topics on all aspects of theoretical computer science. This workshop is an unrefereed meeting. All submissions are accepted for the presentation. There should be no problem of presenting these papers at refereed conferences and/or journals. This meeting is familiar and widely open for everyone who is interested in theoretical computer science. It is held twice a year (January/February and July/August). If you have a chance, I recommend that you attend it. Check the website http://www.dais.is.tohoku.ac.jp/eatcs_japan/ for further details. The list of the presentations is as below; you can see the activity of the Japanese society of theoretical computer science.

Program of the 23rd EATCS-JP/LA workshop on TCS

In the following program, “*” indicates speakers. The number [xxS] means it is given by a student speaker.

- [1S] Efficient Enumeration of Block-Cutpoint Trees
*Mengze Qian, Ryuhei Uehara (*Japan Advanced Institute of Science and Technology*)
- [2] Enumeration of Paths and Minimal Vertex Cuts on Directed Hypergraphs
*Kazuhiro Kurita (*Nagoya University*), Kevin Mann (*University of Trier*)
- [3] On Path Enumeration in Temporal Graphs
*Yu Nakahata (*Nara Institute of Science and Technology*)
- [4S] An Algorithm and Data Structure for Steiner Tree Query
*Kosuke Sugiyama (*Nagoya University*), Tesshu Hanaka (*Kyushu University*), Hirotaka Ono (*Nagoya University*), Kunihiko Sadakane (*The University of Tokyo*)
- [5S] Generating Extended Formulations using Non-Deterministic ZSDD
*Kazuki Takakura (*Kyushu University*), Kohei Hatano (*Kyushu University / RIKEN-AIP*), Eiji Takimoto (*Kyushu University*)
- [6S] Chord Similarity-Based Music Transformation
*Kosei Matsushita, Kazuhiro Kurita, Hirotaka Ono (*Nagoya University*)
- [7S] Induced Subgraph Isomorphism Reconfiguration under Extended Reconfiguration Rules
*Tatsuhiro Suga, Akira Suzuki, Yuma Tamura, Xiao Zhou (*Tohoku University*)
- [8S] Parameterized Complexity of 2-Layer and Outer k -Planarity Testing
Yasuaki Kobayashi (*Hokkaido University*), *Yuto Okada (*Nagoya University*), Alexander Wolff (*Universität Würzburg*)
- [9] Max-Distance Sparsification for Diversification and Clustering
*Soh Kumabe (*CyberAgent*)
- [10] A Simple Maximal Common Subsequence Index of Multiple Strings
*Yoshifumi Sakai (*Tohoku University*)
- [11S] Equivalence of Glaph Regular Expressions with Rewriting Rules and Spanning Tree Automata
*Hitoshi Ino, Akio Fujiyoshi (*Ibaraki University*)
- [12S] Analysis of Specific Boards in Strings-and-Coins
*Souta Kobayashi, Ryo Yoshinaka, Ayumi Shinohara (*Tohoku University*)
- [13S] Shapley Value for Rectified Linear Function Game
*Shunta Yamazaki, Tomomi Matsui (*Institute of Science Tokyo*)
- [14S] Shapley Value for Traveling Salesman Games with the Symmetric Monge Property
*Shota Tsukada, Tomomi Matsui (*Institute of Science Tokyo*)
- [15S] The 5/6-Density Threshold Conjecture for Pinwheel Scheduling with Real-Valued Periods
*Kouta Miyagi, Hiroshi Fujiwara, Katsuhisa Ouchi (*Shinshu University*)
- [16S] The Optimal Question Number for YOMEN
*Kouki Hirano (*Nagoya University*), Hironori Kiya (*Osaka Metropolitan University*), Tesshu Hanaka (*Kyushu University*), Hirotaka Ono (*Nagoya University*)
- [17S] NP-Completeness of Envy-Free and Exchange-Stable Seat Arrangement Problems on Grid Graphs

- **Sota Kawase, Shuichi Miyazaki (University of Hyogo)*
[18S] Solving Numerical Puzzles using Quantum Annealing
 **Chiho Fukunaga, Seiya Okubo (University of Shizuoka)*
[19S] Characteristics of Sorting Algorithms Learned via Deep Reinforcement Learning
 **Kouki Shiga, Kanta Ozawa, Koichi Yamazaki (Tokyo Denki University)*
[20S] Complexity of the Steiner Walk Problem with Vertex-Weight Functions
 **Ryutaro Suzuki, Rin Saito, Takehiro Ito (Tohoku University)*
[21S] A Loopless Algorithm for Generating Gray Code of Binary Trees Encoded in Left-Child Sequences
 **Naoya Okuzumi, Kenji Mikawa (Maebashi Institute of Technology)*
[22S] Dominating Set Gray Code Problem on Complete Split Graphs
 **Kokai Toranosuke, Naoki Domon, Akira Suzuki, Takahiro Suzuki, Yuma Tamura, Xiao Zhou (Tohoku University)*
[23] The Convergence Rate of Computable Predictions
 **Kenshi Miyabe (Meiji University, Department of Mathematics)*
[24] On Parameters Causing the Hardness of the Matrix Compression Problem
 **Dominik Köppl (University of Yamanashi), Vincent Limouzy (University Clermont Auvergne), Andrea Marino (University of Florence), Giulia Punzi (University of Pisa), Takeaki Uno (National Institute of Informatics)*
[25] Dudeney's Dissection is Optimal
 *Erik D. Demaine (Massachusetts Institute of Technology), *Tonan Kamata, Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*
[26S] Edge Overlap-freeness of Prisms with a Continuous Parameter
 *Tonan Kamata (Japan Advanced Institute of Science and Technology), Jason S. Ku (National University of Singapore), *Takumi Shiota (Kyushu Institute of Technology), Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*
[27S] Dissections of a Net of a Regular Octahedron into Nets of Regular Octahedra
 **Yuta Nomi (Japan Advanced Institute of Science and Technology), Takumi Shiota (Kyushu Institute of Technology), Tonan Kamata, Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*
[28S] Construction of Common Unfolding by Fixed-Point Propagation Method
 **Taiga Goto, Tonan Kamata, Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*
[29] Finding Distinct 2-Maximal Independent Sets is Hard
 **Yasuaki Kobayashi (Hokkaido University), Kazuhiro Kurita (Nagoya University)*
[30S] A Fast Algorithm for the Cartesian Tree Isomorphism Problem with k Mismatches
 **Yoshihiro Kotaki, Souta Kobayashi, Ryo Yoshinaka, Ayumi Shinohara (Tohoku University)*
[31S] Dynamic Programming Algorithm for Lot-Sizing Problem with Variable Production Capacity
 **Yu Osawa, Akiyoshi Shioura (Institute of Science Tokyo)*
[32S] An Improved Analysis of Non-Clairvoyant Makespan Minimization Scheduling with Predictions
 **Naoto Kaneko, Hiroshi Fujiwara, Katsuhisa Ouchi (Shinshu University)*
[33S] Online Bin Packing Algorithms for Two Item Sizes

- **Kiryu Furuya, Hiroshi Fujiwara, Katsuhisa Ouchi (Shinshu University)*
- [34] Self-Stabilizing Graph Exploration by a Single Agent
 **Yuichi Sudo (Hosei University), Sayaka Kamei (Hiroshima University), Fukuhito Ooshita (Fukui University of Technology)*
- [35S] A Note on the Treewidth of Toroidal Grids
 *Tatsuji Gima (Hokkaido University), *Hiraku Morimoto, Yuto Okada, Yota Otachi (Nagoya University)*
- [36S] Additive Sensitivity of Dictionary Based Compression and String Attractors
 **Yuto Fujie, Hiroki Shibata, Yuto Nakashima, Shunsuke Inenaga (Kyushu University)*
- [37] On Burrows-Wheeler Transform and Dictionary Compression
 **Hideo Bannai (Institute of Science Tokyo), Tomohiro I (Kyushu Institute of Technology), Yuto Nakashima (Kyushu University)*
- [38S] Pattern Matching on Run-Length Grammar-Compressed Strings in Linear Time
 **Yuto Iguchi, Ryo Yoshinaka, Ayumi Shinohara (Tohoku University)*

Past/Forthcoming Events

WALCOM 2025 & 2026

International Conference and Workshops on Algorithms and Computation (WALCOM) conference has been established to encourage the researchers of theoretical computer science in Asia, especially, India and Bangladesh. Nowadays, there are many participants from a wide range of Asia, not so many from Europe so far. The organizers give a big welcome to many attendees from Europe. The 19th WALCOM (WALCOM 2025) was held in Chengdu, China, from February 28th to March 2nd, 2025. See <https://tcsuestc.com/walcom2025> for more information on WALCOM 2025. The next WALCOM will be held in Perugia, Italy. This is the first WALCOM in Europe.

AAAC 2025

Annual Meeting of the Asian Association for Algorithms and Computation (AAAC) aims at promoting collaborations in theoretical computer science in Asia (but not restricted in the region). The 16th AAAC was held in Hong Kong, on May 31st and June 1st, 2025. See <https://conference.cs.cityu.edu.hk/aaac2025/> for more information on AAAC 2025.

CG Week 2025

The Computational Geometry Week (CG Week) is the premier international forum for advances in computational geometry and its many applications. CG Week combines a number of events, most notably the 41st International Symposium on Computational Geometry (SoCG 2025), the associated Media Exposition (CG:ME), workshops, the Young Researchers Forum (CG:YRF), and the CG Challenge (CG:SHOP). The 2025 edition will be held in Kanazawa, Japan, from June 23rd to 27th, 2025. See <https://socg25.github.io/index.html> for more information on CG Week 2025.

WAAC 2025

The 25th Japan-Korea Joint Workshop on Algorithms and Computation (WAAC 2025) will be held in Sapporo, Japan, on August 19th and 20th, 2025. The aim of this workshop

The Bulletin of the EATCS

is to provide a forum for researchers working on algorithms and the theory of computation, to promote the exchange of recent results, to foster new collaborations among researchers. Historically, the workshop has established for the purpose of collaboration of researchers of Korea and Japan; however, participation from any country is welcome. See <https://waac-alg-comp.github.io/2025/> for more information of WAAC 2025.

ISAAC 2025

International Symposium on Algorithms and Computation (ISAAC) is intended to provide a forum for researchers working on algorithms and computation. The 36th edition of this symposium will be held in Tainan, Taiwan, from December 7th to 10th, 2025. See <https://isaac2025.csie.ncku.edu.tw/> for more information on ISAAC 2025.

Submission Deadline: June 30, 2025 (Anywhere on Earth)

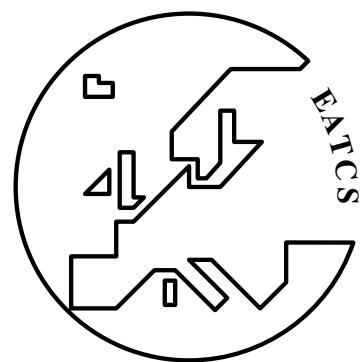
Notification of Acceptance: August 30, 2025

EATCS JAPAN CHAPTER

CHAIR: RYUHEI UEHARA
VICE CHAIR: TAKEHIRO ITO
SECRETARY: YUTO NAKASHIMA
EMAIL: EATCS-JP@GRP.TOHOKU.AC.JP
URL: [HTTP://WWW.DAIS.IS.TOHOKU.AC.JP/EATCS_JAPAN/](http://WWW.DAIS.IS.TOHOKU.AC.JP/EATCS_JAPAN/)

BEATCS no 146

European
Asso**c**i**a**n**t**ion f**o**r
Theoretical
Computer
Science



E **A** **T** **C** **S**
106

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual International Colloquium on Automata, Languages and Programming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Award of research and academic career prizes, including the EATCS Award, the Gödel Prize (with SIGACT), the Presburger Award, the EATCS Distinguished Dissertation Award, the Nerode Prize (joint with IPEC) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

Benefits offered by EATCS to Young Researchers also include:

- Database for Phd/MSc thesis
- Job search/announcements at Young Researchers area

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologna, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014
- Kyoto, Japan 2015
- Rome, Italy 2016
- Warsaw, Poland 2017
- Prague, Czech Republic 2018
- Patras, Greece 2019
- Saarbrücken, Germany (virtual conference) 2020
- Glasgow, UK (virtual conference) 2021
- Paris, France 2022
- Paderborn, Germany 2023
- Tallinn, Estonia 2024

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D. theses;
- Entertainments and pictures related to computer science.

The Bulletin of the EATCS

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at <http://www.eatcs.org/bulletin>. Questions and proposals can be addressed to the Editor by email at bulletin@eatcs.org.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the EATCS Monographs and Texts are now M. Henzinger (Vienna), J. Hromkovič (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:

*Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands*

who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies. The Editor-in-Chief of the journal currently are D. Sannella (Edinburgh), L. Kari and P.G. Spirakis (Patras).

ADDITIONAL EATCS INFORMATION

For further information please visit <http://www.eatcs.org>, or contact the President of EATCS:

*Prof. Giuseppe F. Italiano,
Email: president@eatcs.org*

EATCS MEMBERSHIP

DUES

The dues are €40 for a period of one year (two years for students / Young Researchers). Young Researchers, after paying, have to contact secretary@eatcs.org, in order to get additional years. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €35 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional €35 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website www.eatcs.org, where you will find an online registration form and the possibility of secure online payment. Alternatively, contact the Secretary Office of EATCS:

*Mrs. Efi Chita,
Computer Technology Institute & Press (CTI)
1 N. Kazantzaki Str, University of Patras campus,
26504, Rio, Greece
Email: secretary@eatcs.org,
Tel: +30 2610 960333, Fax: +30 2610 960490*

If you are an EATCS member and you wish to prolong your membership or renew the subscription you have to use the Renew Subscription form. The dues can be paid via paypal and all major credit cards are accepted.

For additional information please contact the Secretary of EATCS:

*Dmitry Chistikov
Computer Science
University of Warwick
Coventry
CV4 7AL
United Kingdom
Email: secretary@eatcs.org,*
